ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Ανάπτυξη Εφαρμογής σε Android για τη διαχείριση χρηστών με βάση τη γεωγραφική τοποθεσία τους

Πτυχιακή εργασία

ΜΗΤΡΟΠΟΥΛΟΣ ΙΩΑΝΝΗΣ

Αθήνα, Μάιος 2017

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Τριμελής Εξεταστική Επιτροπή

Τσερπές Κωνσταντίνος
Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Βαρλάμης Ηρακλής
Επίκουρος Καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Αναγνωστόπουλος Δημοσθένης
Καθηγητής,
Τμήμα Πληροφορικής και Τηλεματικής,
Χαροκόπειο Πανεπιστήμιο

Ο Μητρόπουλος Ιωάννης

δηλώνω υπεύθυνα ότι:

1) Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

2) Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ τον κ.Κωνσταντίνο Τσερπέ για την βοήθειά του και τις εύστοχες υποδείξεις του κατά τη διάρκεια της πτυχιακής μου εργασίας.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως στόχο την ανάπτυξη μιας εφαρμογής σε περιβάλλον Android η οποία δίνει τη δυνατότητα σε κάθε χρήστη να αναφέρει τη γεωχωρική τοποθεσία του ώστε να μπορεί να βρίσκει τους κοντινούς προς αυτόν χρήστες. Εν συνεχεία, υπάρχει η δυνατότητα επικοινωνίας μεταξύ αυτών μέσω γραπτών μηνυμάτων.

Για την καταχώριση της τοποθεσίας του χρήστη σε μια απομακρυσμένη βάση δεδομένων έγινε μελέτη και σύγκριση μεταξύ αλγορίθμων τοποθέτησης αντικειμένων σε κατανεμημένα συστήματα αποθήκευσης όπως είναι η MongoDB. Το βασικό κριτήριο στη σύγκρισή τους αποτέλεσε ο χρόνος επιστροφής των διαθέσιμων χρηστών. Ο αριθμός των χρηστών ήταν μερικές δεκάδες χιλιάδες.

Λέξεις κλειδιά: android application, mongo, γεωγραφική τοποθεσία, geohashing αλγόριθμοι

Abstract

The goal of this thesis is the development of a software application in Android environment that allows each user to apply his geographical location so he can find the nearby users. Then, there is the ability to communicate with them via text messaging.

In order to register user's location in a remote data base was made a study and a comparison between object placement storage algorithms in distributed storage systems such as MongoDB. The main criterion in this comparison was the return time of the available users. The number of users was a few tens of thousands.

Keywords: android application, mongo, geographic area, geohashing algorithms

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

ΚΑΤΑΛΟΓΟΣ ΔΙΑΓΡΑΜΜΑΤΩΝ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

app	application
OS	Operating System
GPS	Global Positioning System
DBMS	Data Base Management System
XMPP	Extensible Messaging and Presence Protocol
SDK	Software Development Kit
IoT	Internet of Things

ΠΡΟΛΟΓΟΣ

Στην πτυχιακή μου εργασία, έφτιαξα μια ολοκληρωμένη εφαρμογή από την αρχή, η οποία αποτελείται από τρία ξεχωριστά τμήματα: μια βάση και δύο προγράμματα, ένα backend και ένα frontend.
Η βάση σχεδιάστηκε σε μια MongoDB.
Το backend αποτελείται από Restful Web Services με τη χρήση του Java Framework Spring Boot.

Το frontend είναι μία απλή Android εφαρμογή με την οποία μπορούν οι χρήστες της να επικοινωνούν με τους κοντινούς τους μέσω μηνυμάτων. Λόγω της αυξημένης χρήσης εφαρμογών, η εφαρμογή θα πρέπει να μπορεί να διαχειρίζεται μεγάλο αριθμό χρηστών. Με βάση αυτή την υπόθεση, διαχειριζόμαστε τα δεδομένα ως “μεγάλα δεδομένα” (big data) και καταφεύγουμε σε αντίστοιχες λύσεις για την εκτέλεση πειραμάτων. Για την καλύτερη αξιοποίηση αυτών, ώστε να μην υπάρχουν ανεπιθύμητες καθυστερήσεις και να επιστρέφονται σωστά, θα δείξουμε ότι χρειάζεται να κατανεμηθούν αποδοτικά στη βάση. Η καταλληλότερη λύση φαίνεται να είναι η χρήση ενός κατανεμημένου συστήματος διαχείρισης των δεδομένων, και η βέλτιστη τοποθέτηση των δεδομένων λαμβάνοντας υπόψη τη γεωχωρική τους φύση.
Για την εύρεση του τρόπου της αποδοτικής τους κατανομής στη βάση, δοκιμάστηκαν πειραματικά τρεις μέθοδοι-αλγόριθμοι τοποθέτησης αντικειμένων στη κατανεμημένη βάση MongoDB. Το κριτήριο για να διαπιστωθεί η απόδοση αποτέλεσε το response time για ένα ερώτημα εύρεσης των πλησιέστερων χρηστών. Οι αλγόριθμοι που χρησιμοποιήθηκαν ήταν: αυτός που χρησιμοποιεί η MongoDB, ο αλγόριθμος EHDP που σχεδιάστηκε από το Πανεπιστήμιο του Πεκίνου και η απλή μέθοδος εισαγωγής δεδομένων στη βάση. Η τελευταία, αποτέλεσε και το σημείο αναφοράς για την αξιοπιστία των άλλων.

Αρχικά, γίνεται μια μικρή εισαγωγή για την αναγκαιότητα και περιγραφή των τεχνολογιών και των εργαλείων που χρησιμοποιήθηκαν, δίνοντας έμφαση στις έννοιες των Big data και στον τρόπο λειτουργίας των NoSQL βάσεων, όπως επίσης και σε αλγόριθμους τοποθέτησης αντικειμένων σε κατανεμημένες βάσεις. Ύστερα, αναφέρονται παρόμοιες εφαρμογές και οι διαφορές τους από αυτή που ανέπτυξα. Κατόπιν, αναλύονται η αρχιτεκτονική και τα βήματα για την ανάπτυξη της εφαρμογής. Στη συνέχεια, αναλύονται τα αποτελέσματα της μελέτης των αλγορίθμων που χρησιμοποιήθηκαν για την εισαγωγή των δεδομένων στη βάση τέλος, παρουσιάζονται εικόνες της τελικής εφαρμογής από διάφορα σενάρια χρήσης και προτείνονται κάποιες μελλοντικές της επεκτάσεις.

ΚΕΦ.1: Εισαγωγή

1.1. Γενικά

Τα τελευταία χρόνια, η συνεχής πρόοδος της τεχνολογίας άλλαξε τον τρόπο που οι άνθρωποι αλληλοεπιδρούν μεταξύ τους αλλά και με το γύρο περιβάλλον τους. Σήμερα, υπάρχει η δυνατότητα σε οποιονδήποτε χρήστη, κυρίως μέσω των κοινωνικών δικτύων και των blog, να δημοσιεύει στο διαδίκτυο μεγάλο όγκο δεδομένων. Ο μεγάλος αυτός όγκος, καθιστά δύσκολη την επεξεργασία και την εξαγωγή γνώσης από αυτόν. Όμως, με τον σωστό τρόπο αποθήκευσης αυτών η επεξεργασία τους γίνεται ευκολότερη. Επιπλέον, υπάρχει η τάση της διεπαφής του χρήστη με το περιβάλλον του. Αυτό σημαίνει ότι ο χρήστης χρειάζεται δεδομένα που σχετίζονται με την τοποθεσία του. Δηλαδή, έχουμε τεράστιο όγκο δεδομένων που έχουν γεωχωρική διάταξη τα οποία πρέπει να επεξεργάζονται και να διατίθενται άμεσα στους χρήστες.

Μας ενδιαφέρει να γνωρίζουμε τη θέση των αντικειμένων ανά πάσα στιγμή, αφού η τοποθεσία τους αποτελεί το κύριο στοιχείο της εφαρμογών που καλύπτουν την παραπάνω ανάγκη. Τέτοιες εφαρμογές χρησιμοποιούνται σε διάφορες τεχνολογίες και υπηρεσίες, όπως για παράδειγμα σε IoT, κινητές συσκευές, drones, ακόμα και στο στρατό.
Παράδειγμα εφαρμογής του geolocation σε ένα σύστημα IoT είναι για την αποφυγή συγκρούσεων. Η χρήση τους σε mobile εφαρμογές είναι αρκετά συχνή, ωστόσο χαρακτηριστικό παράδειγμα αποτελούν εφαρμογές που προτείνουν τοποθεσίες-σημεία ενδιαφέροντος ανάλογα με τη γεωγραφική περιοχή του χρήστη. Επίσης, στα drones υπάρχει αυτή η ανάγκη ώστε να γνωρίζουμε που βρίσκονται, ιδιαίτερα όταν μεταφέρουν κάποιο φορτίο. Τέλος, υπάρχουν στρατιωτικές εφαρμογές που παρακολουθούν τη πορεία των πλοίων ή των αεροσκαφών του ναυτικού και της αεροπορίας αντίστοιχα. Είναι, επίσης, σημαντικό να μπορεί να προβλεφθεί η τροχιά τους.

Συμπερασματικά, επειδή ο όγκος δεδομένων είναι τεράστιος, υπάρχει αυξημένη ανάγκη τέτοιων συστημάτων που υποστηρίζουν big data storage και spatial placement για την βέλτιστη αξιοποίησή τους.

1.2. Προκλήσεις

Παρ’όλη την ανάγκη για τέτοια συστήματα δεν έχουν αναπτυχθεί ακόμα πολλές εφαρμογές. Αυτό συμβαίνει γιατί οι προκλήσεις που καλούμαστε να αντιμετωπίσουμε δεν είναι εύκολα υλοποιήσιμες. Λόγω του μεγάλου όγκου δεδομένων θεωρείται αναγκαία η χρήση μη σχεσιακών κατανεμημένων βάσεων δεδομένων. Επίσης, για την σωστή λειτουργία τους, πρέπει να γίνεται με ακρίβεια και προσοχή η τοποθέτησή τους σε αυτές για να μπορέσει το σύστημα να

αξιοποιήσει τις δυνατότητες της NoSQL βάσης, δηλαδή την αποδοτική κατανομή των αντικειμένων σε αυτή. Η αποδοτική κατανομή προϋποθέτει ότι τα αντικείμενα έχουν τη διάσταση του χώρου. Συνεπώς, γίνεται εύκολα αντιληπτό ότι τέτοια συστήματα έχουν μεγάλη πολυπλοκότητα.

Η τοποθέτηση δεδομένων σχετίζεται με το πρόβλημα της βέλτιστης διανομής των αντικειμένων μεταξύ των sever. Οι τρόποι τοποθέτησης δεδομένων χωρίζονται σε δύο κατηγορίες: σε γεωγραφικής τοποθέτησης, όπου τα δεδομένα μετακινούνται μεταξύ των κέντρων δεδομένων που αντιπροσωπεύουν μια γεωγραφική περιοχή, και σε καταμερισμού δεδομένων, στις οποίες τα δεδομένα μετακινούνται μεταξύ των κόμβων.[1]

1.3. Τρόπος Αξιολόγησης

Για την αντιμετώπιση αυτών των προκλήσεων στην εφαρμογή που δημιουργήθηκε για την παρούσα πτυχιακή εργασία, έγιναν πειράματα σχετικά με τον τρόπο που οι spatial placement αλγόριθμοι επιδρούν στην απόδοση του συστήματος. Τα πειράματα δοκιμάστηκαν με διάφορα σενάρια χρήσης σε μια μη κατανεμημένη βάση δεδομένων, την MongoDB. Η απόδοση του συστήματος καθορίστηκε από το response time ερωτημάτων που έγιναν στη βάση, δηλαδή το πόσο γρήγορα εξυπηρετείται ο χρήστης.

1.4. Δομή κειμένου

Στις επόμενες ενότητες, περιγράφονται αναλυτικά τα πειράματα μαζί με τους αλγόριθμους που έγιναν καθώς επίσης και η παρουσίαση-ανάλυση των αποτελεσμάτων τους.

Πιο συγκεκριμένα, στην ενότητα 2, παρουσιάζεται η τρέχουσα κατάσταση των τεχνολογιών τόσο για την αποθήκευση μεγάλου όγκου δεδομένων όσο και παρόμοιων messaging εφαρμογών για κινητά τηλέφωνα, αναφέροντας ενδεικτικά παραδείγματα για κάθε περίπτωση.
Στην ενότητα 3, γίνεται μια σύντομη περιγραφή της εφαρμογής, εξηγώντας παράλληλα τον σκοπό του πειράματος που πραγματοποιήθηκε. Επιπλέον, γίνεται παρουσίαση των ερωτημάτων που έγιναν στη βάση ώστε να αξιολογηθούν οι αλγόριθμοι τοποθέτησης δεδομένων. Τέλος, γίνεται μια αναλυτική περιγραφή της λειτουργίας αυτών των αλγορίθμων.
Στην ενότητα 4, αναφέρεται το θεωρητικό υπόβαθρο για την δημιουργία μιας ολοκληρωμένης εφαρμογής, καθώς επίσης και τα εργαλεία που χρησιμοποιήθηκαν.
Στην ενότητα 5, αναγράφονται οι απαιτήσεις του συστήματος.
Στην ενότητα 6 , περιγράφεται η αρχιτεκτονική και όλα τα δομικά στοιχεία που συνδέθηκαν ώστε να δημιουργηθεί η εφαρμογή.
Στην ενότητα 7, παρουσιάζονται και αναλύονται τα αποτελέσματα του πειράματος. Επιπλέον, σε αυτή την ενότητα γίνεται η εξαγωγή συμπερασμάτων με βάση τα αποτελέσματα.

Στην ενότητα 8, εμφανίζονται οι οθόνες από διάφορα σενάρια χρήσης της εφαρμογής.

Τέλος, στην ενότητα 9, αναφέρονται μερικές μελλοντικές εργασίες που πιθανόν θα μπορούσαν να βελτιώσουν τα αποτελέσματα των ερωτημάτων, αλλά και την εφαρμογή.

ΚΕΦ.2: Υφιστάμενη κατάσταση

2.1. Big Data

Με τον όρο big data εννοούμε μια συλλογή μεγάλων ή/και πολύπλοκων δεδομένων τα οποία είναι δύσκολο να επεξεργαστούν από παραδοσιακές τεχνικές επεξεργασίας δεδομένων[2]. Συνήθως χρησιμοποιείται για τη περιγραφή προηγμένων μεθόδων που έχουν ως σκοπό την εξαγωγή τιμών από τα δεδομένα. Για τη διαχείριση και ανάλυση των Big Data, μεταξύ άλλων, απαιτείται η χρήση μαθηματικών, στατιστικών, τεχνικών βελτίωσης, επεξεργασίας σημάτων, τεχνικών απεικόνισης και νευρωνικών δικτύων[3].

Δεν χρησιμοποιούνται μόνο στην επιστήμη της πληροφορικής, αλλά αξιοποιούνται σε διάφορους κλάδους όπως της οικονομίας, της πολιτικής και των επιστημών. Για αρκετές δεκαετίες, η οικονομία εξελίχθηκε δίνοντας μεγάλη έμφαση στην εμπειρική εργασία(empirical work). Όμως, η ραγδαία ανάπτυξη των δεδομένων επηρέασε την οικονομική έρευνα. Όλο και περισσότερο, οι οικονομολόγοι χρησιμοποιούν τα πρόσφατα διαθέσιμα διοικητικά δεδομένα ή τα δεδομένα του ιδιωτικού τομέα τα οποία συχνά αποκτώνται μέσω συνεργασιών με ιδιωτικές επιχειρήσεις, δημιουργώντας νέες ευκαιρίες και προκλήσεις[4]. Πρόσφατο παράδειγμα αξιοποίησης των big data στη πολιτική είναι στις Αμερικάνικες Προεδρικές εκλογές του 2016, όπου πολιτικοί αναλυτές προσπάθησαν να αυξήσουν τους ψηφοφόρους με βάση τις πληροφορίες που είχαν για αυτούς, κάνοντας έτσι, την αναζήτηση πιθανών υποστηρικτών ευκολότερη, ταχύτερη και λιγότερο δαπανηρή σε σχέση με τις προηγούμενες φορές[6]. Το Google Scholar είναι μια μηχανή αναζήτησης ελεύθερα προσβάσιμη η οποία ευρετηριάζει το πλήρες κείμενο ή τα μεταδεδομένα της επιστημονικής βιβλιογραφίας. Περιλαμβάνει επιστημονικά περιοδικά και βιβλία, δημοσιεύσεις σε συνέδρια, διατριβές, προεκτυπώσεις, περιλήψεις, τεχνικές εκθέσεις και άλλες επιστημονικές πηγές, συμπεριλαμβανομένων των δικαστικών γνωμών και των διπλωμάτων ευρεσιτεχνίας[6]. Ενώ η Google δεν δημοσιεύει το μέγεθος της βάσης δεδομένων του Google Scholar, εκτιμάται ότι τον Μάιο του 2014 περιείχε περίπου 160 εκατομμύρια έγγραφα[7]. Τα επιστημονικά δεδομένα θεωρούνται big data επειδή η αποθηκευτική και υπολογιστική απαίτηση τους είναι υψηλή ώστε να εξασφαλίζονται μέσω παραδοσιακών αρχιτεκτονικών. Υψηλή απαίτηση έχουν και στην απόδοση των δεδομένων για αυτό χρησιμοποιούνται ειδικά εργαλεία. Τέλος, χρειάζονται ακρίβεια στο φιλτράρισμα των δεδομένων για την σωστή κατηγοριοποίησή και ταξινόμησή τους.

Επίσης, η αποτελεσματική αξιοποίησή των Big Data, μας δίνει τη δυνατότητα να χρησιμοποιούμε υπηρεσίες όπως mobile-banking, ηλεκτρονικό εμπόριο, γρήγορες αναζητήσεις στο διαδίκτυο, γεωχωρικά δεδομένα κα[8].

2.2. NoSQL

Οι σχεσιακές βάσεις δεδομένων (RDBMS) χρησιμοποιούνται ως επί το πλείστων για την αποθήκευση των σημαντικών δεδομένων των οργανισμών ή ως το back-end ενός web service. Ωστόσο, είναι πολύ δαπανηρές, υπολειτουργικές και υπερβολικά πολύπλοκες, ώστε να εφαρμοστούν σε τομείς όπως Big Data Analysis, Ανάλυση Αρχείων, Υπηρεσίες Κοινωνικών Δικτύων και Εφαρμογές Κινητών Τηλεφώνων.

Για την αντιμετώπιση του προβλήματος της αποθήκευσης και ανάκτησης μεγάλου όγκου δεδομένων με βάση τη γεωγραφική τοποθεσία υπάρχουν κατανεμημένες NoSQL βάσεις, αντικαθιστώντας έτσι τις σχεσιακές. Αποτελούν, ουσιαστικά, ένα αποτελεσματικό μέσο αποθήκευσης και πρόσβασης σε big data, επειδή οι server τους είναι πιο εύκολα οριζόντια κλιμακωτοί και αναπαραγόμενοι από τις σχεσιακές DBMS. Έχουν σχεδιαστεί για την διαχείριση χιλιάδων ή εκατομμυρίων χρηστών που εκτελούν απλές λειτουργίες δεδομένων όπως ενημερώσεις και αναγνώσεις[9]. Αυτό το μοντέλο δεδομένων δεν έχει καθορισμένο σχήμα έτσι ώστε οι χρήστες να μπορούν εύκολα να αλλάξουν με δυναμικό τρόπο το μοντέλο δεδομένων των εφαρμογών τους. Αυτά τα χαρακτηριστικά των NoSQL DBMS είναι που τις καθιστούν κατάλληλες και χρησιμοποιούνται όλο και περισσότερο σε ανάλυση σε πραγματικό χρόνο, υπηρεσίες διαδικτύου, κινητές εφαρμογές και αποθήκευση μηχανογραφικών δεδομένων, όπως τα αρχεία καταγραφής και τα δεδομένα IoT (Internet of Things)[10].

Ορισμένες από αυτές έχουν σχεδιαστεί για αποθήκευση γεωγραφικών δεδομένων, ενώ άλλες χρειάζονται κάποιο extension ώστε να την περιλαμβάνουν. Κάποιες άλλες δεν έχουν σχεδιαστεί για γεωγραφικές εφαρμογές, αλλά έχουν εφαρμοστεί για την υποστήριξη γεωχωρικών δεδομένων. Μερικές από τις NoSQL databases που συνηθίζεται να χρησιμοποιούνται για τη διαχείριση γεωχωρικών δεδομένων είναι οι MongoDB (μια ανοιχτού κώδικα κατανεμημένη βάση), BigTable (χρησιμοποιείται στο Google Earth, αναπτύχθηκε από την Google[11]), Cassandra(μια ανοιχτού κώδικα κατανεμημένη βάση, αναπτύχθηκε από την Facebook[12]), CouchDDB (μια ανοιχτού κώδικα κατανεμημένη βάση, αναπτύχθηκε από την Apache[13]). Οι MongoDB και CouchDDB είναι Document database, η BigTable είναι Column Store, ενώ η Cassandra Key-Value store.

2.2.1. Key-Value store

Είναι ένα μοντέλο αποθήκευσης δεδομένων που έχει σχεδιαστεί για την αποθήκευση, την ανάκτηση και τη διαχείριση αντικειμένων ενός λεξικού. Τα λεξικά περιέχουν μια συλλογή αντικειμένων ή αρχείων, τα οποία με τη σειρά τους έχουν πολλά διαφορετικά πεδία με δεδομένα μέσα σε αυτά. Αυτά τα αντικείμενα, αποθηκεύονται και ανακτώνται χρησιμοποιώντας ένα κλειδί που τα προσδιορίζει με μοναδικό τρόπο και χρησιμοποιείται για την γρήγορη εύρεση των δεδομένων μέσα στη βάση δεδομένων.

Οι αποθηκευμένες τιμές αντιστοιχούν σε συγκεκριμένα κλειδιά. Το κλειδί μπορεί να το ορίζει ο χρήστης ή να είναι αυτόματα παραγόμενο ενώ η τιμή μπορεί να είναι οποιοσδήποτε τύπος αντικειμένου.

2.2.2. Column store

Είναι ένα σύστημα διαχείρισης βάσεων δεδομένων (DBMS) που αποθηκεύει πίνακες δεδομένων ανά στήλη και όχι ανά γραμμή. Η αποθήκευση των δεδομένων γίνεται με χρονολογική σειρά, ωστόσο, αποθηκεύοντας δεδομένα σε στήλες και όχι σε γραμμές, η βάση μπορεί να έχει πιο ακριβή πρόσβαση στα δεδομένα που χρειάζεται ώστε να απαντήσει σε ένα ερώτημα παρά να σαρώνει και να απορρίπτει ανεπιθύμητα δεδομένα ανά γραμμή. Τα αποτελέσματα των ερωτημάτων μπορούν να μην περιέχουν όλα τα πεδία της γραμμής, αλλά μόνο αυτά που ζητήθηκαν από το ερώτημα. Επίσης, η αλλαγή μιας τιμής σε κάποια στήλη γίνεται χωρίς να διαβαστεί ολόκληρη η γραμμή στην οποία ανήκει. Ως αποτέλεσμα, η απόδοση των ερωτημάτων αυξάνεται συχνά[14], ιδιαίτερα σε πολύ μεγάλα σύνολα δεδομένων.

Απεικονίζονται με τον ίδιο τρόπο όπως οι σχεσιακές βάσεις, λόγω της σχεδίασης του πίνακα.

2.2.3. Document database

Είναι μια βάση δεδομένων που έχει σχεδιαστεί για την αποθήκευση, την ανάκτηση και τη διαχείριση πληροφοριών σε έγγραφα. Αποθηκεύουν όλες τις πληροφορίες για ένα συγκεκριμένο αντικείμενο ως έγγραφο σε μια μοναδική εγγραφή στη βάση δεδομένων. Τα έγγραφα γίνεται να έχουν διαφορετική δομή για κάθε αντικείμενο. Κάθε αποθηκευμένο αντικείμενο είναι διαφορετικό από κάθε άλλο και διαχωρίζεται από τα υπόλοιπα μέσω ενός μοναδικού Id που αποκτά κατά την εισχώρησή του. Αυτό κάνει την απεικόνιση αντικειμένων στη βάση δεδομένων μια απλή εργασία, εξαλείφοντας υπό φυσιολογικές συνθήκες αντικείμενα με παρόμοιες. Αυτό την καθιστά εύχρηστη για τον προγραμματισμό εφαρμογών οι οποίες μεταβάλλονται συνεχώς και για εφαρμογές στις οποίες η ανάπτυξή τους πρέπει να γίνει γρήγορα, αφού το αντικείμενο μετατρέπεται άμεσα σε έγγραφο.

Τα έγγραφα ενσωματώνουν και κωδικοποιούν δεδομένα (ή πληροφορίες) με κάποια τυπική μορφή ή κωδικοποίηση. Οι κωδικοποιήσεις που χρησιμοποιούνται περιλαμβάνουν τα XML, YAML, JSON και BSON, καθώς και δυαδικές μορφές όπως έγγραφα PDF και Microsoft Office. Αν μια εγγραφή είναι πχ τύπου JSON, τότε μια δεύτερη δεν είναι υποχρεωτικό να είναι JSON, μπορεί για παράδειγμα να είναι XML.

Αποτελούν υποκατηγορία μιας Key-Value store βάσης.

Ο παρακάτω πίνακας παρουσιάζει μερικά από τα βασικά χαρακτηριστικά των βάσεων δεδομένων NoSQL που περιγράφηκαν παραπάνω:

	Εκτέλεση	Ευελιξία	Περιπλοκότητα
Key-Value store	Υψηλή	Υψηλή	Μηδενική
Column Store	Υψηλή	Μέτρια	Χαμηλή
Document database	Υψηλή	Υψηλή	Χαμηλή

Πίνακας1: Σύγκριση των χαρακτηριστικών των τρόπων αποθήκευσης NoSQL βάσεων

2.3. Τεχνικές τοποθέτησης δεδομένων

Είδαμε ότι η τοποθέτηση των αντικειμένων στη βάση αποτελεί τη βασική πρόκληση για εφαρμογές που διαχειρίζονται μεγάλο όγκο δεδομένων σύμφωνα με τη γεωχωρική τοποθεσία του τελικού τους χρήστη. Παρακάτω παρατίθενται οι κύριες τεχνικές που χρησιμοποιούνται σε κατανεμημένες βάσεις δεδομένων για την τοποθέτηση δεδομένων:

Συνεχής κατακερματισμός(Consistent hashing): αυτή η μέθοδος χρησιμοποιεί μια λειτουργία κατακερματισμού για να αντιστοιχίσει ένα κλειδί αντικειμένου σε ένα server. Αρχικά, οι server οργανώνονται μεταξύ τους και τους αποδίδεται ένα μοναδικό αναγνωριστικό. Στη συνέχεια, με τη χρήση μιας συνάρτησης κατακερματισμού, οι τιμές των κλειδιών των αντικειμένων αντιστοιχίζονται σε ένα από τα αναγνωριστικά των server[15].

Διαχωρισμός πολλαπλών χαρακτηριστικών(Multi-Attribute Sharding): η μέθοδος που κατανέμει τα δεδομένα με βάση μόνο το primary κλειδί χωρίς να λαμβάνονται υπόψιν τα άλλα χαρακτηριστικά[16].

Φίλτρα Bloom(Bloom filters): αυτά αποτελούνται από αποτελεσματικές πιθανολογικές δομές δεδομένων και χρησιμοποιούνται για να ελέγξουν αν ένα αντικείμενο είναι τοποθετημένο σε έναν δεδομένο κόμβο.

Πιθανοτικός συσχετιζόμενος πίνακας(Probabilistic Associative Array): αυτή η μέθοδος είναι μια δομή δεδομένων χωρικής αποδοτικής τοποθέτησης και μπορεί να χρησιμοποιηθεί για την αποθήκευση αυθαίρετων συσχετισμών μεταξύ των κλειδιών και των κόμβων. Πολλαπλά φίλτρα Bloom χρησιμοποιούνται για την παρακολούθηση των εισαγόμενων δεδομένων και των ταξινομητών Tree Tree Decision[17].

Δυναμική τοποθέτηση δεδομένων(Dynamic data placement): η αλλαγή της τοποθέτησης ορισμένων ή όλων των αντικειμένων σε σχέση με την αλλαγή του φόρτου εργασίας.

2.4 . Android messaging applications

Η χρήση έξυπνων τηλεφώνων συνεχώς αυξάνεται. Το λειτουργικό σύστημα που χρησιμοποιείται από τα περισσότερα από αυτά είναι το Android. Έχουν αναπτυχθεί αρκετές εφαρμογές για αυτό το λογισμικό που εξυπηρετούν εκατομμύρια χρήστες καθημερινά.

Η κύρια χρήση των κινητών τηλεφώνων είναι η επικοινωνία μεταξύ των ανθρώπων. Πολλές από τις εφαρμογές δίνουν τη δυνατότητα επικοινωνίας στους χρήστες τους χωρίς γεωγραφικούς περιορισμούς. Το γεγονός αυτό μπορεί να προκαλέσει προβλήματα, κυρίως καθυστέρησης, σε χρήστες της ίδιας περιοχής.

Έχουν υλοποιηθεί πολλές εφαρμογές ανταλλαγής μηνυμάτων. Πολλές από αυτές χρησιμοποιούνται από εκατομμύρια χρήστες καθημερινά, όπως είναι για παράδειγμα το Messenger της Facebook ή το Viber. Ενώ κάποιες άλλες, εξίσου γνωστές, έχουν πουληθεί ακόμα και για μερικά δισεκατομμύρια δολάρια. Μία τέτοια περίπτωση αποτελεί το What's up που εξαγοράστηκε από τη Facebook για 19δισεκατομμύρια δολάρια. Οι παραπάνω εφαρμογές καθώς επίσης και άλλες παρόμοιες έχουν ως κύριο σκοπό την άμεση επικοινωνία μεταξύ των ανθρώπων σε όλο τον κόσμο. Η μόνη προϋπόθεση για να επιτευχθεί αυτό είναι μία σύνδεση στο παγκόσμιο δίκτυο.

Παρ'όλα αυτά, η εφαρμογή που ανέπτυξα δίνει τη δυνατότητα της ανταλλαγής μηνυμάτων μεταξύ χρηστών οι οποίοι βρίσκονται κοντά μεταξύ τους, εκμεταλλευόμενη το πλεονέκτημα της μη σχεσιακής κατανεμημένης βάσης MongoDB να εξυπηρετεί πολλούς χρήστες σε γρηγορότερο χρονικό διάστημα από ότι οι σχεσιακές βάσεις σαν την MySQL. Για την ομαλή λειτουργία της εφαρμογής πέρα από σύνδεση στο δίκτυο απαιτείται η καταγραφή της γεωγραφικής τοποθεσίας του χρήστη. Για αυτό το λόγο γίνεται η χρήση του GPS του τηλεφώνου του εκάστοτε χρήστη.

Η εφαρμογή σχεδιάστηκε και υλοποιήθηκε από την αρχή, αν και υπάρχουν μερικές open source messaging εφαρμογές, όπως το Spika. Το Spika δεν είναι τόσο εφαρμογή όσο μια υπηρεσία messaging που μπορεί ο καθένας να προσαρμόσει στον κώδικά του.

ΚΕΦ.3: Περιγραφή Πειράματος

Όπως αναφέρθηκε σε προηγούμενο κεφάλαιο, η λειτουργία των εφαρμογών που καλούνται να διαχειριστούν γεωχωρικά δεδομένα, βασίζεται στο γεγονός της αποδοτικότερης διαθεσιμότητάς τους, η οποία επιτυγχάνεται με την σωστή τοποθέτηση των αντικειμένων στη βάση. Η τοποθέτηση γίνεται από τον spatial placement αλγόριθμο που εφαρμόζεται σε αυτή.

Για την καταγραφή της τοποθεσίας έγινε μελέτη και σύγκριση μεταξύ αλγορίθμων αποθήκευσης τοποθέτησης αντικειμένων για τη MongoDB.

3.1. Στόχος

Θέλουμε να δείξουμε πως το geohashing επιδρά σε ερωτήματα πιθανών εφαρμογών που περιέχουν γεωχωρική διάταξη, δηλαδή πως επηρεάζει το geohashing τα πιθανά ερωτήματα που μπορούν να κάνουν διάφορες εφαρμογές που βασίζονται-χρησιμοποιούν γεωχωρικά δεδομένα στη βάση.

3.2. Μετρικές

Το βασικό κριτήριο για την σύγκριση των αλγορίθμων αποτέλεσε το response time σε συγκεκριμένα ερωτήματα στη βάση(queries). Τα ερωτήματα(queries) που έγιναν ήταν ο χρόνος αποθήκευσης 1000 χρηστών στη βάση, ο χρόνος ανάκτησης όλων των κοντινών χρηστών σε ακτίνα 100 και 10000 μέτρων αλλά και η επιστροφή των 30000 κοντινότερων χρηστών και ο αντίστοιχος για την εισαγωγή ενός χρήστη και την ανάκτηση του ίδιου αριθμού χρηστών.

Τα response time κάθε δοκιμής μετρήθηκε σε millisecond(ms).

3.2.1. Insert query

Ο χρόνος αποθήκευσης πολλών χρηστών στη βάση δεν έχει πρακτική σημασία, ωστόσο μπορεί να δώσει μερικές χρήσιμες πληροφορίες για την αποδοτικότητα του αλγορίθμου. Για αυτό το λόγο επιλέχθηκε μικρός αριθμός καταχωρήσεων. Για παράδειγμα, εάν δύο διαφορετικοί αλγόριθμοι οι οποίοι επιστρέφουν τον ίδιο αριθμό - μεγάλο όγκο δεδομένων σε γρήγορο χρόνο έχοντας μικρή απόκλιση μεταξύ τους, αλλά έχουν μεγάλη διαφορά στο χρόνο αποθήκευσης στη βάση, τότε θεωρούμε πιο αποδοτικό αυτόν με τον καλύτερο χρόνο εγγραφής στη βάση.

3.2.2. Select query

Ο χρόνος ανάκτησης μεγάλου όγκου δεδομένων από μια DBMS θεωρείται ο πιο σημαντικός παράγοντας στον καθορισμό της απόδοσης του placement αλγορίθμου που χρησιμοποιείται. Αυτό συμβαίνει επειδή στη πράξη ο τελικός χρήστης, όχι μόνο της εφαρμογής που αναπτύχθηκε για αυτή τη πτυχιακή εργασία αλλά και των περισσότερων που διαχειρίζονται big data με βάση τη γεωχωρική τοποθεσία, καλείται να διαχειριστεί αρκετά μεγάλο όγκο δεδομένων, από τους κοντινούς προς αυτόν χρήστες, τον οποίο θέλει να τον έχει διαθέσιμο στο συντομότερο δυνατό χρόνο.

3.2.3. Insert & Select query

Το τρίτο και τελευταίο ερώτημα που έγινε ήταν το response time εγγραφής ενός χρήστη και η επιστροφή σε αυτόν ενός συγκεκριμένου αριθμού χρηστών. Αν και αποτελεί το βασικό σενάριο χρήσης τέτοιων εφαρμογών, ο χρόνος εισχώρησης μιας εγγραφής στη βάση, όπως θα δούμε παρακάτω, δεν είναι σταθερός και έτσι δεν μπορεί να θεωρηθεί αξιόπιστος. Το βασικό σενάριο χρήσης τέτοιων εφαρμογών είναι η καταγραφή της τοποθεσίας του εκάστοτε χρήστη σε μια απομακρυσμένη βάση και εν συνεχεία την ενημέρωση προς αυτόν για τους κοντινούς του χρήστες.

Συνεπώς, το ερώτημα αναφοράς αποτέλεσε το δεύτερο ερώτημα, δηλαδή ο χρόνος ανάκτησης μεγάλου αριθμού χρηστών από τη βάση.

Τα αποτελέσματα καθώς και η σύγκριση τους παρουσιάζεται στη συνέχεια στο κεφάλαιο 7.

3.3 Μεθοδολογία

Για τη συλλογή δεδομένων έγινε χρήση τριών διαφορετικών αλγόριθμων τοποθέτησης αντικειμένων σε μια μη σχεσιακή κατανεμημένη βάση δεδομένων. Έτσι, δημιουργήθηκαν τρία διαφορετικά συστήματα, ένα για κάθε αλγόριθμο, στα οποία δοκιμάστηκε ο αντίστοιχος αλγόριθμος. Οι αλγόριθμοι που εφαρμόστηκαν στα συστήματα αυτά είναι το geohashing που προσφέρει η MongoDB[18], ο αλγόριθμος EHDP[19] και η αποθήκευση των δεδομένων στη βάση χωρίς κάποια προεπεξεργασία, η οποία αποτέλεσε τη baseline υποδομή του πειράματος.

3.3.1. Mongo's Geohashing

Μια τιμή geohash είναι μια δυαδική αναπαράσταση της θέσης σε ένα πλέγμα συντεταγμένων. Η MongoDB για την αποθήκευση ενός αντικειμένου ως γεωχωρικού δεν χρησιμοποιεί αρχείο JSON, αλλά GeoJSON. Το GeoJSON είναι μια μορφή για την κωδικοποίηση μιας ποικιλίας γεωγραφικών δομών δεδομένων. Ένα αντικείμενο

GeoJSON μπορεί να αντιπροσωπεύει μια γεωμετρία, ένα χαρακτηριστικό ή μια συλλογή χαρακτηριστικών. Το GeoJSON υποστηρίζει τους ακόλουθους τύπους γεωμετρίας: Point, LineString, Polygon, MultiPoint, MultiLineString, MultiPolygon και GeometryCollection. Οι λειτουργίες του GeoJSON περιέχουν ένα αντικείμενο γεωμετρίας και πρόσθετες ιδιότητες και μια συλλογή χαρακτηριστικών η οποία αντιπροσωπεύει μια λίστα χαρακτηριστικών. Έχει την ίδια δομή με ένα αρχείο JSON αλλά πρέπει να έχει οπωσδήποτε ένα πεδίο με το όνομα type, το οποίο καθορίζει τον τύπο του αντικειμένου και ένα με όνομα coordinates[20]. Το type μπορεί να πάρει ως παράμετρο ένα String από τα “Point”, “MultiPoint”, “LineString”, “MultiLineString”, “Polygon”, “MultiPolygon”, “GeometryCollection”, “Feature” ή “FeatureCollection”. Οι τιμές που παίρνει το πεδίο coordinates αντιπροσωπεύουν τις γεωγραφικές συντεταγμένες και εξαρτώνται από τον τύπο του document. Στην περίπτωσή μας χρησιμοποιήθηκε ο πιο απλός τύπος που είναι το Point[21]. Άρα το GeoJSON είναι της μορφής{ “type”: “Point”, “coordinates”: [38.075144, 22.673491]}.

Τα ευρετήρια είναι ένας κρίσιμος μηχανισμός για τη βελτιστοποίηση της απόδοσης και της επεκτασιμότητας του συστήματος, παρέχοντας παράλληλα ευέλικτη πρόσβαση στα δεδομένα υποστηρίζοντας έτσι την αποτελεσματική εκτέλεση των ερωτημάτων. Αυτό συμβαίνει επειδή ταξινομούν τα δεδομένα του πίνακα με βάση το πεδίο που ορίστηκε. Χωρίς ευρετήρια, θα πρέπει να εκτελεστεί μια σάρωση συλλογής, δηλαδή να σαρωθεί κάθε έγγραφο σε μια συλλογή, για να επιλεγούν εκείνα τα έγγραφα που ταιριάζουν με τη δήλωση ερωτήματος. Εάν υπάρχει κατάλληλο ευρετήριο για ένα ερώτημα, η MongoDB μπορεί να το χρησιμοποιήσει ώστε να περιορίσει τον αριθμό των εγγράφων που πρέπει να επιθεωρήσει. Η ακρίβεια του ευρετηρίου δεν επηρεάζει την ακρίβεια των ερωτήσεων. Οι πραγματικές συντεταγμένες χρησιμοποιούνται πάντα στην τελική επεξεργασία του ερωτήματος. Τα γεωχωρικά ερωτήματα μπορούν να χρησιμοποιούν είτε επίπεδες είτε σφαιρικές γεωμετρίες, ανάλογα με το ερώτημα και τον τύπο του ευρετηρίου που χρησιμοποιείται. Η MongoDB περιλαμβάνει υποστήριξη για πολλούς τύπους δευτερευόντων ευρετηρίων που μπορούν να δηλωθούν σε οποιοδήποτε πεδίο του εγγράφου. Η χρήση των δευτερευόντων ευρετηρίων επιταχύνει την εύρεση αποτελεσμάτων.

Υποστηρίζει δύο τύπους ευρετηρίων για γεωχωρικά δεδομένα, τα ευρετήρια 2dsphere, που υποστηρίζουν μόνο σφαιρικές γεωμετρίες, και τα 2d, τα οποία υποστηρίζουν τόσο επίπεδες όσο και σφαιρικές γεωμετρίες. Η κύρια διαφορά τους είναι ο τρόπος που υπολογίζουν τις αποστάσεις. Ένα 2d ευρετήριο τις υπολογίζει με βάση την επίπεδη γεωμετρία, ενώ το 2dsphere με βάση τη σφαιρική. Σε πολλές περιπτώσεις, το 2dsphere, είναι πιο ακριβές από το 2d επειδή χρησιμοποιεί τον operator $geoNear, ο οποίος επιστρέφει τα έγγραφα ταξινομημένα από το

πλησιέστερο στο πιο απομακρυσμένο με βάση ένα συγκεκριμένο σημείο. Για να αποφευχθεί η ταξινόμηση ολόκληρης της συλλογής με μία κίνηση, ο αλγόριθμος $geoNear αναπτύσσει επανειλημμένα την αναζήτησή του σε απομακρυσμένα διαστήματα, με στόχο να έχει μερικές εκατοντάδες έγγραφα ανά διαστήματα. Η αναζήτηση όλων των εγγράφων σε ένα διάστημα επιτυγχάνεται με την εύρεση μιας κεφαλίδας δείκτη που καλύπτει. Αυτή η κάλυψη διασφαλίζει ότι όλα τα έγγραφα του διαστήματος μπορούν να βρεθούν χρησιμοποιώντας μια σάρωση ευρετηρίου. Τα έγγραφα στην κάλυψη αλλά όχι στο διάστημα φιλτράρονται μετά. Αφού εντοπιστούν όλα τα έγγραφα σε ένα διάστημα, ταξινομούνται και επιστρέφονται. Αυτή η διαδικασία κάλυψης-φιλτραρίσματος και ταξινόμησης επαναλαμβάνεται για κάθε διάστημα κατά τη διάρκεια της επέκτασης της περιοχής αναζήτησης, έως ότου επιτευχθεί όριο ή έχει κοιτάξει ολόκληρη τη βάση. Για να αποφεύγονται επαναλαμβανόμενες σαρώσεις ευρετηρίου, ο $geoNear συγκεντρώνει κάθε έγγραφο στο πλέγμα και απομονώνει τα έγγραφα που δεν βρίσκονται στο διάστημα αναζήτησης που θα χρησιμοποιηθούν σε ένα επόμενο διάστημα αναζήτησης. Δεδομένου ότι έχει ήδη λάβει κάθε έγγραφο σε προηγούμενες σαρώσεις, παρακολουθεί και την περιοχή που καλύπτεται από προηγούμενα πλέγματα για να διασφαλίσει ότι η τωρινή κάλυψη δεν έχει αλληλοεπικαλύψεις. Για να επιτευχθεί αυτό, ο αλγόριθμος διατηρεί μια ένωση των πλεγμάτων που έχουν ήδη επισκεφθεί. Σε κάθε διάστημα, παίρνει τη διαφορά μεταξύ της τρέχουσας κάλυψης και της σύνδεσης των πλεγμάτων που επισκέπτονται. Έτσι, μειώνεται σημαντικά ο χρόνος των σαρώσεων στο ευρετήριο για μεγάλες αναζητήσεις, όμως για μικρά διαστήματα προστίθενται περιττές λήψεις εγγράφων. Αυτό όμως δεν αποτελεί μεγάλο πρόβλημα γιατί στις περισσότερες περιπτώσεις χρήσης του $geoNear γίνονται ερωτήματα τύπου “Point”, τα οποία πάντοτε καλύπτονται από μόνο ένα πλέγμα[22].

Τα πειράματά μας έγιναν με 2dshere ευρετήριο, εκμεταλλεύοντας τα χαρακτηριστικά του $geoNear operator.

3.3.2. EHDP

Χρησιμοποιήθηκε ένας ιεραρχικός αλγόριθμος τοποθέτησης, που ονομάζεται EHDP, βασιζόμενος στο γεωγραφικό πεδίο των δεδομένων για τη κατανομή χωρικών δεδομένων μεγάλου μεγέθους μεταξύ πολλών συσκευών. Χρησιμοποιεί τον αλγόριθμο Min-Max[23] για την κατηγοριοποίηση των αντικείμενων σε κλάσεις. Ο αλγόριθμος βασίζεται στο SIEVE[24] και το Linear Hashing[25] για την ανάθεση δεδομένων μεταξύ των κλάσεων και στην ανάθεση του γεωγραφικού πεδίου των δεδομένων μέσα στη κλάση. Οι SIEVE και Linear Hashing ισχύουν για μια μόνο αποθήκευση και δεν μπορούν να εξασφαλίσουν και αποδοτική και προσαρμοστική

συμπεριφορά. Λόγω αυτής της χαμηλής προσαρμοστικότητας δεν υπολογίζουν σε συνεχή χρόνο τη θέση του αντικειμένου.

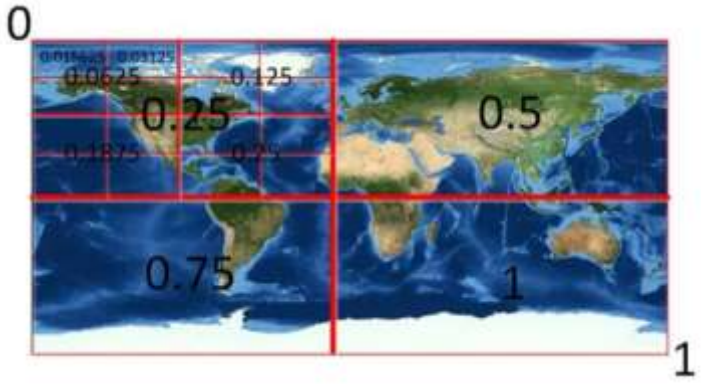
Αντιθέτως, ο EHDP, μπορεί να εντοπίσει τα αντικείμενα σε συνεχή χρόνο γιατί ικανοποιεί την αποδοτικότητα και τη προσαρμοστικότητα. Το σύστημα αποθήκευσης πολλών αντικειμένων αποτελείται από μεγάλη χωρητικότητα κάθε συσκευής και η απόδοση είναι μεγάλη. Για τη διευκόλυνση της διαχείρισης του εξοπλισμού, ο EHDP στην πρώτη χρήση της συσκευής, δεδομένου στις διαφορές της χωρητικότητας εύρους ζώνης της ταξινόμησης, θα τοποθετηθεί σε μια παρόμοια συσκευή του cluster. Επίσης, υποστηρίζει κατανομή βάρους και διάφορα επίπεδα αναπαραγωγής των αντικειμένων. Επιτρέπει δηλαδή, τον προσδιορισμό της αναπαραγωγής στη βάση ανά αντικείμενο κατανέμοντάς το άνισα, ανάλογα το βάρος, για την καλύτερη αξιοποίηση διαφορετικών χαρακτηριστικών απόδοσης για διαφορετικούς server στο σύστημα.

Αρχικά, η συσκευή χρησιμοποιεί έναν clustering αλγόριθμο για να μαζέψει το σύνολο των συσκευών σε μια πληθώρα ομάδων σύμφωνα με το μοντέλο ιεραρχικής συσταδοποίησης για διαχείριση. Η παραπάνω μέθοδος μπορεί να υποστηρίξει την προσθήκη και αφαίρεση συσκευών που δεν μπορούν να απλοποιήσουν τις απαιτήσεις του διαχειριστή ώστε να αποφεύγεται η αντιγραφή των ήδη υπαρχόντων αντικειμένων.
Μετά τη συλλογή του εξοπλισμού χρησιμοποιείται μια αποτελεσματική και ιεραρχικά δίκαιη EFAH Hashing[27] μεταξύ των cluster και ενός clustering αλγόριθμου για την κατανομή των αντικειμένων. Ο αλγόριθμος EFAH Hashing μπορεί να υπολογίσει σε συνεχή χρόνο τη θέση ενός αντικειμένου ώστε ταυτόχρονα να διασφαλιστεί η δίκαιη και καλύτερη αυτόπροσαρμοστικότητα.
Για ένα αντικείμενο x E X θέτουμε μια Hash function h = $\{h_1,h_2,\ldots,h_L\}$, h_i: X→ [0,1]. Αν το διάστημα $h_1(x)$ είναι μέσα σε ένα αντικείμενο του cluster τότε μπαίνει σε αυτό το cluster.
Μόλις το x ανατεθεί σε κάποιο cluster επιλέγουμε τη συσκευή αυτού του cluster.
Έστω ότι ανατέθηκε στο cluster με τη χρήση του γραμμικού D_j, όπου το D_j Hah καθορίζει το x στη συγκεκριμένη θέση, που έχει συσκευές οι οποίες ξεκινάνε στη θέση 0 και φτάνουν στη θέση n_{j-1}. Και έστω ότι το επίπεδο spit της γραμμικής Hash είναι ο δείκτης p με αρχικό επίπεδο το 0. Σύμφωνα με τον τύπο x mod $2^{level} * n_j$ το x ανατίθεται σε μία συσκευή.
Μόλις ανατεθεί στη κατάλληλη συσκευή, τότε επαναϋπολογίζονται τα βάρη των αντικειμένων της συσκευής και γίνεται έλεγχος για τα αντικείμενα που είναι στο ίδιο διάστημα με το x.
Όταν ο αριθμός των clusters g αυξάνεται προκαλεί Interval $2^{[log]+1}$ πρέπει να αυξηθεί ο χώρος μεταξύ κάθε cluster που θα χωριστεί σε δύο κάνοντας το Interval να επεκταθεί στο διπλάσιο του αρχικού.
Σύμφωνα με τις αλλαγές στα βάρη το νέο cluster αυξάνεται. Κατόπιν, προσδιορίζεται το αντικείμενο που έπεσε στην περιοχή σχετικά με την ανάγκη αναδιοργάνωσης των μετακινούμενων αντικειμένων. Συνήθως, οι καινούργιοι servers σε ένα cluster είναι πιο γρήγοροι και με μεγαλύτερη χωρητικότητα από τους ήδη υπάρχοντες. Σε αυτούς θέλουμε να έχουμε περισσότερα αντικείμενα από τους

υπόλοιπους. Για αυτό, χρησιμοποιείται ένας ακέραιος συντελεστής βάρους wj για κάθε cluster. Ο συντελεστής είναι τέτοιος ώστε να δηλώνει την δύναμη του server.

Ύστερα, για το αντικείμενο x με x mod 2 level x nj επαναϋπολογίζονται οι αντίστοιχες συσκευές ώστε να καθορίσουν αν χρειάζεται να μετακινηθεί σε άλλη συσκευή. Τα δεδομένα θα μεταφερθούν στο αντίστοιχο αντικείμενο από τη συσκευή.

Η περιοχή χωρίζεται σε μέρη που αντιστοιχούν στο διάστημα [0,1]. Στη συνέχεια, χρησιμοποιείται η αναδρομική υποδιαίρεση του quadtree σε κάθε γεωγραφικό πεδίο.



Εικονα 1: τρόπος λειτουργίας αλγόριθμου EHDP

Στη περίπτωση που θέλουμε να αποθηκεύσουμε το ίδιο αντικείμενο θα πρέπει να το κάνουμε σε διαφορετικό server και να μην υπάρχει αντίγραφο του ίδιου αντικειμένου σε έναν server.

Τέλος, αξίζει να αναφερθεί ότι για να λειτουργήσει ο αλγόριθμος σε κάθε client απαιτείται ένας πίνακας με όλους τους server του συστήματος, την αποθήκευση των διευθύνσεων δικτύου και μερικά bytes με επιπρόσθετη πληροφορία για τον κάθε server. Αυτό συνεπάγεται ότι χρειάζεται λίγο αποθηκευτικό χώρο και χαμηλές απαιτήσεις συστήματος ώστε να λειτουργήσει σωστά, όπως επίσης και ότι είναι καλός τόσο στην κατανομή των δεδομένων ομοιόμορφα όσο και στην ελαχιστοποίηση της κίνησης των δεδομένων όταν προστίθεται νέα αποθήκευση στο σύστημα.

3.3.3. Baseline υποδομή

Για την τοποθέτηση των αντικειμένων στη βάση δεν χρησιμοποιήθηκε κάποια ειδική τεχνική, ούτε χρησιμοποιήθηκε κάποιο ευρετήριο στη βάση. Η εισχώρηση των δεδομένων έγινε με τον απλό τρόπο που προσφέρει το API της MongoDB για το

οποιοδήποτε αντικείμενο, χωρίς να λαμβάνουμε υπόψιν ότι πρόκειται για γεωχωρικά δεδομένα. Τα αντικείμενα τοποθετούνται το ένα μετά το άλλο.

Αυτός ο τρόπος τοποθέτησης των αντικειμένων στη βάση αποτελεί το βασικό κριτήριο σύγκρισης για την αποδοτικότητα των δύο προηγούμενων.

3.4. Εκτέλεση

Είναι σημαντικό να αναφερθεί ότι όλοι οι αλγόριθμοι εξετάστηκαν με τις ίδιες συνθήκες. Ο κάθε ένας έτρεξε σε διαφορετικό μηχάνημα από τους υπόλοιπους, όμως όλα τα μηχανήματα είχαν τις ίδιες δυνατότητες και χαρακτηριστικά. Επίσης, και οι τρεις επιστρέφουν για κάθε document όλα τα fields που περιέχουν χωρίς να κάνουν κάποιον επιπλέον υπολογισμό. Ο μόνος υπολογισμός που γίνεται είναι η εύρεση των κοντινότερων χρηστών.

ΚΕΦ.4: Θεωρητικό Υπόβαθρο

Για την υλοποίηση μιας εφαρμογής σαν αυτή που δημιουργήθηκε στο πλαίσιο αυτής της πτυχιακής χρειάζονται γνώσεις για τον σχεδιασμό βάσεων δεδομένων, για το χτίσιμο Restful Web Services – API Calls, καθώς επίσης και την κατασκευή της εφαρμογής που αλληλοεπιδράει ο τελικός χρήστης. Για την επίτευξη αυτών υπάρχουν προγράμματα που μας βοηθάνε στην καλύτερη και ταχύτερη ανάπτυξή τους.

4.1. Server

Ο server εξυπηρετεί τις αιτήσεις των άλλων προγραμμάτων(clients). Κάθε πρόγραμμα εκτελεί διαφορετικές λειτουργίες από τα υπόλοιπα, για αυτό υπάρχουν διαφορετικοί τύποι server ώστε να μπορούν να εξυπηρετούν όλα τα προγράμματα ανάλογα τις αιτήσεις τους.

Συνηθισμένοι εξυπηρετητές είναι:

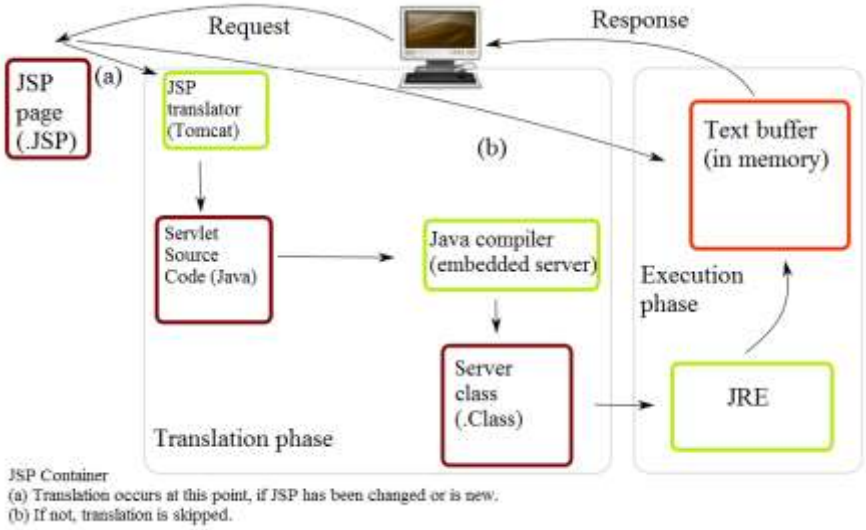
- File Server (εξυπηρετητής αρχείων), για τον διαμοιρασμό αρχείων στο δίκτυο
- Database Server (εξυπηρετητής βάσεων δεδομένων), για την παροχή δεδομένων σε άλλους υπολογιστές
- Proxy Server (εξυπηρετητής διαμεσολαβητή), για την ταχύτερη πλοήγηση στον παγκόσμιο ιστό μειώνοντας την κίνηση στο τοπικό δίκτυο
- Mail Server (εξυπηρετητής ηλεκτρονικού ταχυδρομείου), για την αποστολή και λήψη ηλεκτρονικών γραμμάτων
- HTTP Server, για την επικοινωνία των προγραμμάτων μέσω του HTTP πρωτοκόλλου
- DNS Server, για την αντιστοίχιση των IP διευθύνσεων σε ονόματα
- XMPP Server, για την αποστολή και λήψη μηνυμάτων σε πραγματικό χρόνο αξιοποιώντας το XMPP πρωτόκολλο

4.1.1. Apache Tomcat

Για την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε ο Tomcat server της Apache.

Εικόνα 2: Tomcat Apache Server

Πρόκειται για έναν ανοιχτού κώδικα (open source) server που υλοποιεί Java εφαρμογές, ο οποίος έχει σχεδιαστεί ώστε να επικοινωνεί εύκολα ο java κώδικας με τον server. Στο σχήμα φαίνεται γραφικά πως γίνεται η επικοινωνία με μια jsp σελίδα.



Εικόνα 3: Τρόπος Λειτουργίας Tomcat Apache Server

Επιπλέον, μια από τις ιδιαιτερότητες που έχει είναι το γεγονός ότι περιλαμβάνει Web Socket, τα οποία αναλαμβάνουν την αμφίδρομη επικοινωνία μεταξύ εξυπηρετητή και πελάτη μέσω του TCP πρωτοκόλλου.

Τέλος, σε αυτόν μπορούν να ενσωματωθούν σχεσιακές αλλά και κατανεμημένες βάσεις δεδομένων, όπως η MongoDB που χρησιμοποιήθηκε για τα πειράματα και στην ανάπτυξη της εφαρμογής.

4.1.2. XMPP Protocol

Είναι ένα ανοιχτό πρωτόκολλο βασισμένο στην XML(Extensible Markup Language) το οποίο προσφέρει ανταλλαγή μηνυμάτων στο διαδίκτυο σε πραγματικό χρόνο.

Εικόνα 4: XMPP Protocol

Το πρωτόκολλο μεταφοράς του XMPP είναι το Transmission Control Protocol (TCP). Αυτό σε συνδυασμό με το γεγονός ότι γίνεται πρακτικά stream ένα XML το κάνει βαρύ και όχι και τόσο φιλικό στη μπαταρία των κινητών συσκευών. Έτσι, αποφάσισα να μη το χρησιμοποιήσω για την ανταλλαγή των μηνυμάτων της εφαρμογής αλλά στη θέση του χρησιμοποιήθηκε απευθείας TCP πρωτόκολλο, χωρίς όμως να υπάρχει η δυνατότητα για συνομιλία σε πραγματικό χρόνο, αλλά μόνο on-demand.
Οι περισσότερες messaging εφαρμογές χρησιμοποιούν το XMPP Protocol.

4.2. Βάση Δεδομένων

Η βάση δεδομένων είναι μια οργανωμένη διακριτή συλλογή σχετιζόμενων δεδομένων με συγκεκριμένη δομή ώστε να είναι εύκολα προσπελάσιμη από έναν υπολογιστή που αναζητάει κάποια πληροφορία μέσα σε αυτή.

Για να μπορούν να αλληλοεπιδρούν με τους χρήστες υπάρχουν ειδικά λογισμικά, τα Database Management System (DBMS). Έχουν σχεδιαστεί με τέτοιο τρόπο ώστε να είναι εύκολο στον χρήστη τους να εκτελεί διαδικασίες διαχείρισης, δημιουργίας, ερωτήσεων κα.

Οι πρώτες βάσεις που δημιουργήθηκαν είναι σχεσιακές, ενώ οι πιο πρόσφατες είναι μη σχεσιακές κατανεμημένες. Στις πρώτες, τα δεδομένα αποθηκεύονται σε πίνακες και κάθε ένας περιέχει την απαραίτητη πληροφορία σε γραμμές και στήλες. Πιο συγκεκριμένα, οι στήλες αποτελούν το είδος πληροφορίας ενώ οι γραμμές την πληροφορία. Όλοι οι πίνακες βρίσκονται στον ίδιο υπολογιστή-server. Αντιθέτως, οι κατανεμημένες βάσεις βρίσκονται σε διαφορετικούς υπολογιστές και εκμεταλλεύονται τη δυνατότητα της παραλληλίας μεταξύ των επεξεργαστών, οι οποίοι επικοινωνούν μέσω του διαδικτύου. Αξίζει να σημειωθεί ότι αν και χρησιμοποιούνται πολλοί υπολογιστές, μια πληροφορία αποθηκεύεται σε περισσότερους από έναν, για λόγους διαθεσιμότητας. Επίσης, αυτού του είδους οι βάσεις, δεν έχουν πίνακες αλλά συλλογές. Οι αντίστοιχες στήλες των σχεσιακών βάσεων σε αυτές είναι αρχεία που περιέχουν την πληροφορία ως πεδία.
Κατά την ανάπτυξη της εφαρμογής χρησιμοποιήθηκε μια κατανεμημένη βάση δεδομένων.

4.2.1. MongoDB

Η κατανεμημένη βάση που χρησιμοποιήθηκε είναι η MongoDB, μία open-source JSON-like document-oriented βάση δεδομένων γραμμένη σε C++.

Εικόνα 5: MongoDB

Είναι μια cross-platform(μπορεί να χρησιμοποιηθεί από όλα τα διαφορετικά λειτουργικά συστήματα) βάση που παρέχει υψηλή απόδοση και διαθεσιμότητα. Δουλεύει με συλλογές και έγγραφα και όχι πίνακες και σειρές όπως οι σχεσιακές βάσεις. Τα έγγραφα περιλαμβάνουν σύνολα ζευγών κλειδιού-τιμών και αποτελούν τη βασική μονάδα δεδομένων στη MongoDB. Οι Συλλογές περιέχουν σύνολα των εγγραφών και λειτουργούν ως το ισοδύναμο των πινάκων των σχεσιακών βάσεων δεδομένων.

Όπως και οι άλλες NoSQL βάσεις δεδομένων, η MongoDB υποστηρίζει δυναμική σχεδίαση του σχήματος, επιτρέποντας στα έγγραφα μιας συλλογής να έχουν διαφορετικά πεδία και δομές. Η βάση χρησιμοποιεί μια μορφή αποθήκευσης εγγράφου και ανταλλαγής δεδομένων, την BSON, η οποία παρέχει μια δυαδική αναπαράσταση JSON αρχείων. Παρά το γεγονός ότι είναι απλή στη δομή της, μπορεί να υποστηρίξει σύνθετα ερωτήματα(queries), κάτι το οποίο την καθιστά τόσο ισχυρή όσο η SQL.

Ένα από τα στοιχεία που την κάνουν απλή είναι ότι δεν χρειάζεται η χρήση κάποιου γραφικού περιβάλλοντος όπως το phpmyadmin, αφού τα ερωτήματα μπορούν να γίνουν μέσα από τη γραμμή εντολών-cmd για windows, ή από ένα τερματικό-terminal για unix συστήματα(linux, mac).

4.3. Backend

Το backend είναι το κομμάτι της εφαρμογής που αλληλοεπιδρά με τα δεδομένα. Δεν είναι ορατό από τον χρήστη της εφαρμογής και είναι υπεύθυνο για τον διαμοιρασμό της απαραίτητης πληροφορίας στην εφαρμογή. Επίσης, εδώ γίνονται και πολλοί υπολογισμοί, ώστε να σερβίρεται αυτό που επιθυμεί εφαρμογή χωρίς να χρειαστεί να κάνει πράξεις.

Συνήθως, το backend τμήμα της εφαρμογής δεν βρίσκεται στον ίδιο υπολογιστή με το frontend, αλλά σε κάποιον απομακρυσμένο server. Με αυτόν τον τρόπο ανεξαρτητοποιούμε τον server από τον client.

Πρακτικά, στο backend υλοποιούνται τα Web Services τα οποία καλεί ο client. Με την κλήση ενός Web Service ο client περιμένει μια απάντηση από το backend. Η απάντηση είναι με τη μορφή κειμένου XML, JSON, HTML κτλ, αλλά και με ένας κωδικό, γνωστό ως HTTP Status Code. Πρόκειται για έναν τριψήφιο αριθμό που ενημερώνει τον client σχετικά με την κλήση που έκανε. Για μια επιτυχημένη κλήση και απάντηση λαμβάνει κωδικό 2xx. Για μία επιτυχημένη κλήση αλλά χωρίς να λάβει απάντηση του επιστρέφεται κωδικός 5xx. Σε περίπτωση που ο server δεν μπορεί να παραλάβει την κλήση επιστρέφει κωδικό 4xx. Επίσης υπάρχουν και οι κωδικοί 1xx(πληροφοριακή απάντηση) και 3xx(ανακατεύθυνση κλήσης) οι οποίοι εμφανίζονται σπάνια.

Το backend υλοποιήθηκε με τη χρήση του Spring Boot framework.

Εικόνα 6: Spring Boot Framework

Το Spring Boot είναι ένα βασισμένο στη Java framework για τη δημιουργία εφαρμογών διαδικτύου. Χρησιμοποιεί maven dependencies, δηλαδή μπορεί να αξιοποιήσει εύκολα και γρήγορα όλες τις βιβλιοθήκες της java χωρίς να χρειάζεται να τις εγκαταστήσει τοπικά ο προγραμματιστής, αλλά αρκεί να τις δηλώσει σε ένα ειδικό XML αρχείο, το pom. Εύκολη, όπως επίσης απλή και ασφαλής, θεωρείται και η επικοινωνία με την βάση οποιουδήποτε τύπου αφού γίνεται μέσω JPA και Hibernate. Το Hibernate είναι το ενδιάμεσο λογισμικό που αναλαμβάνει τον σχεδιασμό των Java κλάσεων σε πίνακες της βάσης, καθώς επίσης και την σωστή εφαρμογή των ερωτημάτων στη βάση.

Το εργαλείο που χρησιμοποιήθηκε για την ανάπτυξη κώδικα στο backend είναι το STS Spring του Eclipse.

Εικόνα 7: Spring Tool

4.4. Frontend

Το frontend, ονομάζεται το κομμάτι της εφαρμογής που αλληλοεπιδρά με τον χρήστη. Είναι αυτό που βλέπει ο χρήστης. Περιέχει το γραφικό περιβάλλον το οποίο μπορεί να είναι κείμενο, εικόνα, κουμπιά, μπάρες αναζήτησης και οτιδήποτε άλλο είναι απαραίτητο να βλέπει ο χρήστης ώστε να έχει μια πιο ευχάριστη και εύχρηστη λειτουργία της εφαρμογής.

Μια frontend εφαρμογή συνήθως είναι μια Desktop εφαρμογή, ο χρήστης την εγκαθιστά στον υπολογιστή του, ή μια Web-based, ο χρήστης δεν χρειάζεται να την εγκαταστήσει και είναι προσπελάσιμη από έναν web browser, ή μια mobile, είναι οι εφαρμογές που τρέχουν σε κινητές συσκευές όπως είναι τα smartphones, τα tablets κα.

Το frontend της εφαρμογής που αναπτύχθηκε είναι ένα mobile application. Για την υλοποίησή του έγινε η χρήση SDK για Java. Το SDK είναι μια συλλογή εργαλείων που επιτρέπουν τη δημιουργία εφαρμογών για υπολογιστικά συστήματα, λειτουργικά συστήματα, frameworks κα. Χρησιμοποιούνται συχνά στη κατασκευή κινητών εφαρμογών τόσο σε Android όσο και σε iOS. Επιλέχθηκε η χρήση του Java SDK γιατί η Java είναι μία από τις επίσημες γλώσσες για το Android αλλά και επειδή στο backend χρησιμοποιήθηκε ένα Java framework.

Το Android είναι ένα λειτουργικό σύστημα για κινητές συσκευές που αναπτύχθηκε από τη Google. Είναι σχεδιασμένο για συσκευές με οθόνη αφής(smartphones, tablets), τηλεοράσεις, ρολόγια χειρός, αλλά και αυτοκίνητα.

Εικόνα 8: Android Icon

Η ίδια η Google έχει αναπτύξει ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE) για να διευκολύνει τους προγραμματιστές που θέλουν να αναπτύξουν εφαρμογές στην πλατφόρμα Android. Το περιβάλλον ονομάζεται Android Studio και είναι βασισμένο στο λογισμικό της JetBrains IntelliJ IDEA.

Εικόνα 9: Android Studio IDE

ΚΕΦ.5: Απαιτήσεις

Η εφαρμογή που δημιουργήθηκε είναι μια απλή εφαρμογή επικοινωνίας χρηστών μέσω ανταλλαγής μηνυμάτων. Όταν ο χρήστης ανοίγει την εφαρμογή καταγράφεται η τοποθεσία του μέσω του GPS της συσκευής του και έχει την δυνατότητα να στείλει μηνύματα στους συνδεδεμένους χρήστες που είναι κοντά του σε ακτίνα 100 μέτρων, ενώ ταυτόχρονα στέλνει σε κάποιον τυχαίο από αυτούς ένα κείμενο που έχει επιλέξει. Το κείμενο μπορεί να το επιλέξει από το profile του. Στο profile του έχει τη δυνατότητα να αλλάξει και το όνομά του. Για λόγους οικονομίας των δεδομένων κινητής αλλά και εξοικονόμησης της μπαταρίας της συσκευής του δεν ειδοποιείται για την λήψη ενός νέου μηνύματος αλλά μόνο όταν το επιλέξει ο ίδιος πατώντας ένα ειδικό κουμπί refresh. Τέλος, μετά από ένα χρονικό διάστημα που μια συνομιλία παραμένει ανενεργή τα μηνύματά της διαγράφονται.
Για την χρήση της εφαρμογής δεν απαιτείται εγγραφή του χρήστη, αλλά αρκεί να δηλώσει το username με το οποίο θα είναι εμφανής στους υπόλοιπους.

Για τον σωστό σχεδιασμό και υλοποίηση της εφαρμογής ορίσθηκαν οι απαιτήσεις για αυτό. Οι απαιτήσεις διαχωρίζονται σε Λειτουργικές και Μη Λειτουργικές ανάλογα τον τρόπο που διαχειρίζονται τις παρεχόμενες υπηρεσίες. Έτσι, στις Λειτουργικές απαιτήσεις δηλώνουμε ποιες υπηρεσίες θα πρέπει να παρέχει το σύστημα , πως θα αντιδρά σε συγκεκριμένες εισόδους και πως θα πρέπει να συμπεριφέρεται σε συγκεκριμένες καταστάσεις, ενώ στις Μη Λειτουργικές, δηλώνονται οι περιορισμοί στις υπηρεσίες και στις λειτουργίες που προσφέρει το σύστημα. Τέτοιοι περιορισμοί μπορεί να είναι χρονικοί, αναπτυξιακοί, διάφορα πρότυπα κτλ.

5.1. Λειτουργικές Απαιτήσεις

5.1.1. Καταχώριση τοποθεσίας

Για να είναι εφικτή η επικοινωνία του χρήστη με τους υπόλοιπους χρειάζεται να κοινοποιήσει την τοποθεσία του σε μία βάση ανάλογα με αυτή.

5.1.2. Αυτόματη αποστολή μηνύματος

Μόλις καταχωρηθεί η τοποθεσία του χρήστη τότε επιλέγεται τυχαία ένας άλλος συνδεδεμένος και του στέλνει ένα μήνυμα που έχει επιλέξει.

5.1.3. Αποστολή μηνύματος

Ο χρήστης μπορεί να πληκτρολογεί το μήνυμα που επιθυμεί να στείλει στο ειδικό text area και πατώντας το κουμπί της αποστολής το στέλνει στον επιθυμητό δέκτη.

5.1.4. Αλλαγή ονόματος

Ο χρήστης αλλάζει το όνομα με το οποίο είναι εμφανής στους υπόλοιπους.

5.1.5. Αλλαγή αυτόματου μηνύματος

Ο χρήστης αλλάζει το μήνυμα που θα στέλνεται αυτόματα σε κάποιον τυχαίο μόλις καταχωρηθεί η τοποθεσία του.

5.1.6. Επιλογή συνδεδεμένου χρήστη

Στον χρήστη είναι φανερή μια λίστα με τους συνδεδεμένους κοντινούς χρήστες. Έχει την δυνατότητα να επιλέξει κάποιον από αυτούς ώστε να δει τα μηνύματά τους ή να στείλει κάποιο καινούργιο.

5.1.7. Ανανέωση

Με το κουμπί Refresh ο χρήστης ανανεώνει την τοποθεσία του στη βάση, στέλνει εκ'νέου το επιλεγμένο μήνυμα σε έναν τυχαίο συνδεδεμένο χρήστη και λαμβάνει τόσο τους συνδεδεμένους χρήστες όσο και τα νέα μηνύματα που του έχουν σταλθεί.

Οι παραπάνω αποτελούν Λειτουργικές απαιτήσεις για το frontend κομμάτι της εφαρμογής. Στη συνέχεια αναφέρονται και οι αντίστοιχες για το backend, οι οποίες είναι εξίσου σημαντικές όμως δεν γίνονται αντιληπτές από τον χρήστη.

5.1.8. Καταχώρηση χρήστη σε server

Το γεγονός ότι η βάση της εφαρμογής είναι η MongoDB, δηλαδή μια κατανεμημένη συνεπάγεται ότι είναι εγκατεστημένη σε διαφορετικούς υπολογιστές. Έτσι καλούμαστε να αποφασίσουμε σε ποιον από αυτούς θα αποθηκευτεί. Η απόφαση εξαρτάται από την τοποθεσία του χρήστη. Χρήστες της ίδιας περιοχής αποθηκεύονται στην ίδια βάση.

5.1.9. Λήψη κοντινών χρηστών

Το τμήμα του backend της εφαρμογής που φροντίζει ώστε να επιστρέφει τους χρήστες που είναι κοντά στον αιτούντα. Ανάλογα το σενάριο χρήσης πρέπει να επιστρέφει και τον ανάλογο αριθμό χρηστών(διαφορετικός αριθμός χρηστών θα επιστραφούν για το σενάριο των 30000 πιο κοντινών χρηστών σε σχέση με το αντίστοιχο που επιστρέφει όλους τους χρήστες που είναι κοντά του σε ακτίνα 100 μέτρων).

5.1.10. Διαγραφή μηνυμάτων

Τα μηνύματα διαγράφονται από τον server αν μια συνομιλία παραμείνει ανενεργή για περισσότερο από 2 ώρες.

5.2. Μη Λειτουργικές Απαιτήσεις

5.2.1. Εύρεση Τοποθεσίας

Για να μπορέσει η εφαρμογή να καταγράψει την τοποθεσία του χρήστη θα πρέπει να είναι ανοιχτό το GPS της συσκευής του.

5.2.1. Σωστή Καταγραφή Τοποθεσίας

Για να λειτουργήσει σωστά η εφαρμογή δεν αρκεί μόνο η εύρεση της τοποθεσίας του χρήστη, αλλά πρέπει ο spatial placement αλγόριθμος να την αποθηκεύσει αποδοτικά στη βάση.

5.2.2. Επικοινωνία frontend με backend

Για να μπορούν να αποστέλνονται δεδομένα από το backend στο frontend και αντίστροφα είναι αναγκαία η σύνδεση του χρήστη στο διαδίκτυο.

5.2.3. Απόδοση

Οι χρήστες των εφαρμογών, έχουν την απαίτηση της γρήγορης και αξιόπιστης λειτουργίας τους. Για αυτό το λόγο, κατά τον σχεδιασμό της εφαρμογής καλείται ο προγραμματιστής να φροντίσει ώστε να αποκρίνεται άμεσα χωρίς να καθυστερεί. Για την καλύτερη απόδοσή της, χρησιμοποιήθηκε ο καλύτερος αλγόριθμος spatial

placement που δοκιμάστηκε, όπως προκύπτει από τα αποτελέσματα του πειράματος που περιγράφονται σε επόμενο κεφάλαιο.

Όμως, πρέπει να σημειωθεί ότι κάθε συσκευή έχει διαφορετικές δυνατότητες τις οποίες πρέπει η εφαρμογή να τις ικανοποιεί όλες.

5.2.4. Χρηστικότητα

Το ιδανικό για μια εφαρμογή είναι να δείχνει απλή-εύκολη στον χειρισμό, ώστε να μπορεί να χρησιμοποιηθεί από περισσότερους ανθρώπους που ίσως να μην είναι εξοικειωμένοι με τη τεχνολογία.

Έτσι σχεδιάστηκε και η εφαρμογή που αναπτύχθηκε, η οποία περιλαμβάνει λίγες οθόνες και λειτουργίες.

5.2.5. Ασφάλεια

Στις μέρες μας, η ηλεκτρονική ασφάλεια είναι κάτι που απασχολεί την πλειοψηφία των χρηστών. Ένας τρόπος για την ασφαλή χρήση μιας εφαρμογής είναι η ανωνυμία, την οποία την επιδιώκει η πλειοψηφία των χρηστών.

Η εφαρμογή αυτή, διασφαλίζει ανωνυμία καθώς για την χρήση της δεν χρειάζεται κάποια εγγραφή, παρά μόνο η δήλωση ενός ψευδωνύμου, το οποίο μπορεί να το αλλάζει κάθε χρήστης όσες φορές θέλει.

ΚΕΦ.6: Αρχιτεκτονική

Παρακάτω περιγράφονται όλα τα δομικά στοιχεία που συνδέθηκαν ώστε να δημιουργηθεί η εφαρμογή.

6.1. Βάση Δεδομένων

Η εφαρμογή έχει ως βάση μια MongoDB, η οποία περιλάμβανε δύο Συλλογές, τις ‘users’ και ‘msg’. Στη Συλλογή ‘users’ είναι καταχωρημένα όλα τα απαραίτητα στοιχεία των χρηστών ώστε να λειτουργήσει σωστά η εφαρμογή, ενώ στη Συλλογή ‘msg’ αποθηκεύονται προσωρινά τα μηνύματα που ανταλλάζουν οι χρήστες.

Αναλυτικότερα,
η Συλλογή ‘users’ έχει 7 Πεδία:

- id: κάθε χρήστης μόλις συνδεθεί στη βάση αποκτά ένα id. επίσης, είναι το primary key του πίνακα. προστίθεται από τη MongoDB αυτόματα.
- title: είναι το όνομα του Document, είναι προαιρετικό πεδίο.
- username: αποτελεί το όνομα του χρήστη που είναι ορατό στους άλλους.
- lat: αναπαριστά τη τοποθεσία του γεωγραφικού πλάτους στο οποίο βρίσκεται ο χρήστης, όπως καταγράφηκε από το GPS του.
- lon: αναπαριστά τη τοποθεσία του γεωγραφικού μήκους στο οποίο βρίσκεται ο χρήστης, όπως καταγράφηκε από το GPS του.
- message: είναι το προεπιλεγμένο μήνυμα κάθε χρήστη, το οποίο στέλνεται σε έναν τυχαίο χρήστη.
- lastlogin: είναι ένα timestamp που δηλώνει τη χρονική στιγμή που ενημέρωσε την τοποθεσία του ο χρήστης.

ενώ, η Συλλογή ‘msg’ έχει 4 Πεδία:

- message: είναι το μήνυμα που στάλθηκε.
- sender: εκφράζει τον αποστολέα του μηνύματος.
- receiver: εκφράζει τον παραλήπτη του μηνύματος.
- created: δηλώνει τη χρονική περίοδο που στάλθηκε το μήνυμα.

Η δομή του Document της Συλλογή ‘users’ διαφέρει ανάλογα τον spatial placement αλγόριθμο που εφαρμόστηκε. Οι παρακάτω εικόνες παρουσιάζουν τη μορφή του JSON ανά περίπτωση.
Οι αλλαγές που υπάρχουν βασίζονται στο γεγονός ότι κάθε αλγόριθμος έχει διαφορετικό τρόπο αποθήκευσης, ώστε να εκμεταλλευτεί τις δυνατότητές του και να προσφέρει το καλύτερο αποτέλεσμα.

```
{
    "_id" : ObjectId("9b13c5ddfd55a0abdc1838751"),
    "title" : "User_0",
    "type" : "Feature",
    "properties" : {
        "username" : "User_0",
        "message" : " my name is User_0"
        "lastlogin" : {
            "$date" : 1496939962837
        }
    },
    "geometry" : {
        "type" : "Point",
        "coordinates" : [37.9611591, 23.7089022]
    }
}
```


Εικόνα 10: MongoDB's placement Document


```
{
    "_id" : ObjectId("5eabd5eab33875abcdef10547"),
    "title" : "User_16411",
    "loc" : {
        "lng" : 37.9611591,
        "lat" : 23.7089022
    },
    "username" : "User_16411",
    "message" : " my name is User_16411"
    "lastlogin" : {
        "$date" : 1496939976957
    }
}
```


Εικόνα 11: EHDP's placement Document


```
{
    "_id" : ObjectId("594d5eab3f54474fd785783e"),
    "title" : "user_1",
    "username" : "user_1",
    "lat" : 37.9611591,
    "lon" : 23.7089022,
    "message" : "Hi, I am user_1",
    "lastlogin" : {
        "$date" : 1496939954162
    }
}
```


Εικόνα 12: Baseline placement Document

Στην Συλλογή ‘msg’ τα Πεδία sender και receiver αντιπροσωπεύουν ένα username από την Συλλογή ‘users’.

6.2. Backend

Στο backend αναπτύχθηκαν τα REST Web Services ώστε να επικοινωνεί με τη frontend εφαρμογή και με τη βάση καθώς επίσης και οι αλγόριθμοι για τους απαραίτητους υπολογισμούς.

6.2.1. Models

Τα models αναλαμβάνουν να διαχειριστούν τα δεδομένα της εφαρμογής. Πρακτικά είναι τα αντικείμενα του προγράμματος.

Τα μοντέλα που αναπτύχθηκαν είναι:

- Location: έχει πεδία τα lat και lon, τα οποία είναι τύπου double και χρησιμοποιούνται για την αποθήκευση της τοποθεσίας του χρήστη, τόσο το γεωγραφικό πλάτος(lat), όσο και το γεωγραφικό μήκος(lon).
- User: έχει πεδία τα username και message που είναι τύπου String, ενώ παράλληλα κληρονομεί και από το Location. Το username είναι το όνομα του χρήστη και το message είναι το επιλεγμένο μήνυμα που θα σταλθεί σε έναν τυχαίο χρήστη.
- Message: έχει πεδία τα message, sender, receiver και mine. Τα τρία πρώτα είναι String ενώ το mine είναι Boolean. Το message είναι το περιεχόμενο του μηνύματος, τα sender και receiver δείχνουν τον αποστολέα και τον παραλήπτη, ενώ το mine δηλώνει αν το μήνυμα είναι του τωρινού χρήστη.

6.2.2. Controllers

Στους Controllers υλοποιήθηκαν τα Web Services με τα οποία γίνεται η επικοινωνία του backend με το frontend.

Οι κλήσεις αυτές είναι:

- apply: καλείται με μέθοδο POST, παίρνει ως είσοδο JSON και παράγει JSON.
  Το JSON που παίρνει ως είσοδο είναι ένας User.
  Το JSON που παράγει είναι μία λίστα με τους συνδεδεμένους χρήστες. Για τον κάθε χρήστη επιστρέφεται και μία λίστα με τα μηνύματα μεταξύ αυτού και των συνδεδεμένων.
- sendMessage: καλείται με μέθοδο POST, παίρνει ως είσοδο JSON και παράγει JSON.
  Το JSON που παίρνει ως είσοδο περιλαμβάνει έναν user και ένα message.

Φροντίζει ώστε να στείλει το μήνυμα του message. Αφού σταλθεί το μήνυμα τότε καλείται η apply.
Το JSON που παράγει είναι το αποτέλεσμα της κλήσης apply.

- getUsers: καλείται με μέθοδο GET, παίρνει ως ορίσματα latitude και longitude και επιστρέφει όλους τους χρήστες που είναι σε ακτίνα 100μέτρων από αυτό το σημείο.
- deleteUsers: αναλαμβάνει να διαγράψει όλους τους χρήστες που ήταν ανενεργοί για περισσότερες από δύο ώρες.

Όλες οι κλήσεις που περιγράφηκαν παραπάνω καλούνται από το frontend εκτός από την τελευταία(deleteUsers), η οποία καλείται από ένα cron job κάθε 3 ώρες.

6.2.3. Clients

Στους Clients γίνεται η επικοινωνία με τη βάση όπως επίσης υλοποιήθηκαν οι αλγόριθμοι ώστε να επιστρέφονται τα αναμενόμενα από το frontend αποτελέσματα.

Πιο συγκεκριμένα οι μέθοδοι που το πετυχαίνουν αυτό είναι:

- applyLocation: αναλαμβάνει την επικοινωνία με τη βάση. Παίρνει ως παράμετρο ένα User και επιστρέφει τους διαθέσιμους χρήστες και τα μηνύματά τους με τον αιτούντα.
  Την καλεί η κλήση apply.
- applyUser: αποθηκεύει στη βάση τον χρήστη που παίρνει ως παράμετρο.
  Την καλεί η μέθοδος applyLocation.
- getOnlineUsers: παίρνει ως παράμετρο έναν χρήστη και επιστρέφει τους διαθέσιμους σε αυτόν χρήστες.
  Την καλεί η μέθοδος applyLocation αφού ολοκληρωθεί με επιτυχία η applyUser.
- sendMessageToRandomUser: στέλνει το επιλεγμένο μήνυμα σε έναν τυχαίο χρήστη, από τους διαθέσιμους, με τον οποίο δεν έχει ανταλλάξει κανένα μήνυμα. Στη συνέχεια, επιστρέφει όλα τα μηνύματα που έχει με τους υπόλοιπους.
  Την καλεί η μέθοδος applyLocation αφού ολοκληρωθεί με επιτυχία η getOnlineUsers.
- sendMessage: παίρνει ως παράμετρο ένα μήνυμα και το στέλνει στον receiver.
  Την καλεί η μέθοδος sendMessageToRandomUser μόλις επιλέξει τον τυχαίο χρήστη. Επίσης καλείται από την ομώνυμη κλήση.
- getMessages: παίρνει ως παράμετρο έναν χρήστη και επιστρέφει όλα τα διαθέσιμα μηνύματα του.
  Την καλεί η μέθοδος sendMessageToRandomUser μόλις στείλει το μήνυμα στον τυχαίο χρήστη.
- deleteUsers: αναλαμβάνει να διαγράψει όλους τους χρήστες που ήταν ανενεργοί για περισσότερες από δύο ώρες.

Την καλεί η ομώνυμη κλήση.

- deleteMessages: για κάθε χρήστη που διαγράφεται στη deleteUsers διαγράφει τα μηνύματά του.
  Την καλεί η deleteUsers όταν διαγράψει τους ανενεργούς χρήστες.
- isNear: είναι η μέθοδος που δέχεται ως παράμετρο έναν χρήστη και φροντίζει να φέρει τους χρήστες που είναι κοντά του σε ακτίνα των 100 μέτρων. Χρησιμοποιείται μόνο στη περίπτωση χρήσης της baseline υποδομής.
- init: ανάλογα την τοποθεσία του χρήστη φροντίζει να τον τοποθετήσει στον σωστό server.
  Καλείται στον constructor της κλάσης πριν ξεκινήσει κάποια άλλη ενέργεια.

6.3. Frontend

Στο frontend δημιουργήθηκαν όλες οι διεπαφές που χρειάζεται ο χρήστης για την αποτελεσματική λειτουργία της εφαρμογής, όπως επίσης και οι TCP/IP κλήσεις ώστε να επικοινωνεί με τη backend εφαρμογή και να διαχειρίζεται τα δεδομένα που λαμβάνει από αυτή.

6.3.1. Βιβλιοθήκες

Για την σωστή λειτουργία, αλλά και την πιο γρήγορη υλοποίηση της εφαρμογής, χρησιμοποιήθηκαν οι εξής βιβλιοθήκες:

- apache commons io: βιβλιοθήκη για λειτουργείες εισόδου-εξόδου
- gson: βιβλιοθήκη για την δημιουργία και διαχείριση JSON

6.3.2. Manifest

Είναι ένα XML αρχείο που παρέχει σημαντικές πληροφορίες για μια Android εφαρμογή. Αν δεν δηλωθούν κάποιες πληροφορίες δεν θα τρέξει η εφαρμογή, ενώ κάποιες άλλες είναι προαιρετικές. Αναλυτικότερα, τα tags μαζί με την πληροφορία τους που χρησιμοποιήθηκαν στην εφαρμογή είναι:

- manifest ✍ είναι το root tag του αρχείου και περιέχει όλα τα άλλα tags.
- uses-permission → σε αυτό το tag δηλώνονται τα δικαιώματα της συσκευής που δίνουμε στην εφαρμογή. Χρησιμοποιήθηκαν τα:
  - INTERNET: η άδεια που δίνουμε στην εφαρμογή να χρησιμοποιεί το διαδίκτυο
  - ACCESS_FINE_LOCATION: η άδεια που δίνουμε στην εφαρμογή να χρησιμοποιεί το GPS και να μας δίνει ακριβή τοποθεσία
- Application → σε αυτό το tag δηλώνονται πληροφορίες της εφαρμογής, τα activities και τα services που χρησιμοποιούνται από αυτή.

6.3.3. Activities

Τα activities είναι οι οθόνες της εφαρμογής. Οι οθόνες που δημιουργήθηκαν είναι:

- MainActivity: είναι η SplashScreen της εφαρμογής, δηλαδή η οθόνη που εμφανίζεται καθώς φορτώνει το άνοιγμα της και εκκινεί την βασική.
- HomeActivity: είναι η βασική οθόνη της εφαρμογής, ξεκινάει με το άνοιγμα της και αποτελείται από:
  - τη λίστα με τα μηνύματα: σε αυτή τη λίστα εμφανίζονται όλα τα μηνύματα του χρήστη με τον αντίστοιχο που συνομιλεί
  - τη λίστα με τους διαθέσιμους χρήστες: σε αυτή τη λίστα εμφανίζονται όλοι οι διαθέσιμοι χρήστες
  - το refresh button: είναι το κουμπί ώστε ο κάθε χρήστης να ενημερώνει την τοποθεσία του και να λαμβάνει νέα μηνύματα on demand
  - ένα textarea και ένα sendbutton: το textarea είναι η περιοχή στην οποία ο χρήστης γράφει το μήνυμά του και με το sendbutton το στέλνει στον επιθυμητό χρήστη
- NavigationDrawerActivity: είναι η συρόμενη πλευρική οθόνη της εφαρμογής, η οποία αποτελείται από:
  - το όνομα του χρήστη, μαζί με ένα κουμπί το οποίο του επιτρέπει να το αλλάξει. το προεπιλεγμένο όνομα κάθε χρήστη είναι Guest
  - ένα textarea και ένα button, τα οποία είναι κρυμμένα και εμφανίζονται μόνο όταν ο χρήστης πατήσει το κουμπί αλλαγής ονόματος
  - στη θέση των παραπάνω textarea και button είναι εμφανής ένα μήνυμα κατάστασης του χρήστη. όταν ο χρήστης είναι έχει internet σύνδεση και έχει καταγραφεί η τοποθεσία του από το GPS είναι available, αλλιώς αναγράφεται το αντίστοιχο μήνυμα σφάλματος. μόλις εμφανίζονται τα παραπάνω κρύβεται αυτό
  - ένα κουμπί settings, το οποίο δίνει τη δυνατότητα στο χρήστη να αλλάξει το μήνυμα που θα στέλνει σε έναν τυχαίο χρήστη
  - ένα κουμπί about, το οποίο περιέχει κάποιες πληροφορίες σχετικά με την εφαρμογή
- DefaultMessageActivity: η οθόνη στην οποία ο χρήστης μπορεί να αλλάξει το προεπιλεγμένο μήνυμά του. αποτελείται από ένα text το οποίο εμφανίζει το τωρινό και από ένα textarea και ένα button με τα οποία ορίζει ένα νέο.
- AboutActivity: αποτελείται από ένα κείμενο με πληροφορίες της εφαρμογής.

6.3.4. Adapters

Είναι το view που εμφανίζει και διαχειρίζεται τα αντικείμενα των λιστών.

Οι adapters που χρησιμοποιήθηκαν είναι:

- AvailableUsersListAdapter: είναι ο adapter που περιλαμβάνει τη λίστα με τους διαθέσιμους χρήστες και για κάθε χρήστη έχει τα μηνύματά του, ώστε να εμφανίζει τον αριθμό τους δίπλα από το όνομά του. Στη περίπτωση που η λίστα είναι άδεια φροντίζει να εμφανίζει σχετικό μήνυμα.
- ChatAdapter: είναι ο adapter που περιλαμβάνει τη λίστα με τα μηνύματα που πρέπει να εμφανίζονται. Αναλαμβάνει την εμφάνισή τους με τρόπο τέτοιο ώστε να είναι αριστερά της οθόνης τα μηνύματα του χρήστη της εφαρμογής και δεξιά του χρήστη με τον οποίο συνομιλεί. Επίσης, όταν ο χρήστης επιλέγει κάποιον άλλο μέσω του AvailableUsersListAdapter πρέπει να εμφανίζει τα σωστά μηνύματα, δηλαδή αυτά που έχει με αυτόν.

6.3.5. Models

Τα models αναλαμβάνουν να διαχειριστούν τα δεδομένα της εφαρμογής. Πρακτικά είναι τα αντικείμενα του προγράμματος.

Τα μοντέλα που αναπτύχθηκαν είναι:

- Location: έχει πεδία τα lat και lon, τα οποία είναι τύπου double και χρησιμοποιούνται για την αποθήκευση της τοποθεσίας του χρήστη, τόσο το γεωγραφικό πλάτος(lat), όσο και το γεωγραφικό μήκος(lon).
- User: έχει πεδία τα username και message που είναι τύπου String, ενώ παράλληλα κληρονομεί και από το Location. Το username είναι το όνομα του χρήστη και το message είναι το επιλεγμένο μήνυμα που θα σταλθεί σε έναν τυχαίο χρήστη.
- Message: έχει πεδία τα message, sender, receiver και mine. Τα τρία πρώτα είναι String ενώ το mine είναι Boolean. Το message είναι το περιεχόμενο του μηνύματος, τα sender και receiver δείχνουν τον αποστολέα και τον παραλήπτη, ενώ το mine δηλώνει αν το μήνυμα είναι του τωρινού χρήστη.
- ClientReponse: είναι το περιεχόμενο του JSON αρχείου που λαμβάνονται από το backend. Έχει πεδία τα status, httpCode, message, dataUsers, dataMessages.
    - status: είναι τύπου String και περιέχει την κατάσταση του δικτύου του χρήστη. Οι τιμές που μπορεί να λάβει είναι OK, για την επιτυχημένη σύνδεση του χρήστη στο διαδίκτυο, NO_NETWORK_CONNECTION, για την περίπτωση που ο χρήστης δεν έχει σύνδεση στο network και NO_INTERNET_CONNECTION, εάν ο χρήστης δεν έχει σύνδεση στο internet.
    - httpCode: είναι τύπου int και επιστρέφει το HTTP Code της κλήσης που πραγματοποιήθηκε.

- message: είναι τύπου String και περιέχει την επεξήγηση του httpCode.
- dataUsers: είναι μία λίστα τύπου User η οποία περιέχει τους Users που επιστρέφει η βάση.
- dataMessages: είναι μία λίστα τύπου Message η οποία περιέχει τα Messages που επιστρέφει η βάση.

6.3.6. Controllers

Στους Controllers υλοποιήθηκαν οι listeners ώστε να είναι λειτουργική η εφαρμογή.

Οι listeners αυτοί είναι:

- applyLocation: ενεργοποιείται με την εκκίνηση της εφαρμογής. Μόλις πάρει τη τοποθεσία του χρήστη από το GPS καλεί τον client apply και περιμένει την απάντησή του. Αν η σύνδεση με το backend έγινε με επιτυχία, δηλαδή το clientResponse status είναι OK, και υπάρχουν διαθέσιμοι χρήστες στέλνονται στον αντίστοιχο adapter ώστε να τους εμφανίσει στη λίστα του HomeActivity. Σε περίπτωση που το response status του client δεν είναι OK τότε τυπώνεται σε ένα Toast με σχετικό μήνυμα σφάλματος.
- sendMessage: ενεργοποιείται όταν ο χρήστης πατήσει το sendMessage button. Παίρνει το περιεχόμενο του text area και καλεί τον sendMessage client ώστε να σταλθεί το μήνυμα στον επιλεγμένο χρήστη. Αφού σταλθεί με επιτυχία τότε ο ChatAdapter το εμφανίζει στη λίστα των μηνυμάτων.
- showAvailableUsers: καλείται από τον applyLocation controller με σκοπό να εμφανίσει τους συνδεδεμένους χρήστες.

6.3.7. Clients

Στους Clients υλοποιήθηκαν οι TCP/IP κλήσεις με τις οποίες γίνεται η επικοινωνία του frontend με το backend. Οι Clients καλούνται από τους Controllers.

Οι κλήσεις αυτές είναι:

- apply: παίρνει την τοποθεσία του χρήστη και την στέλνει στο backend, μέσω της POST κλήσης apply. Από αυτό λαμβάνει ένα αντικείμενο ClientReponse.
- getUsers: παίρνει ως όρισμα τη τοποθεσία του χρήστη και επιστρέφει από το backend, μέσω της GET κλήσης getUsers, ένα αντικείμενο ClientReponse, μόνο με τους χρήστες και όχι τα μηνύματά τους.
- sendMessage: παίρνει ως όρισμα ένα μήνυμα και το στέλνει στο backend, μέσω της POST κλήσης sendMessage και επιστρέφει μήνυμα επιτυχίας ή αποτυχίας της αποστολής του.

6.3.7. Utilities

Τα utilities είναι μικρά τμήματα κώδικα που βοηθούν στην αξιοποίηση διεργασιών του συστήματος από την εφαρμογή.

Για την λειτουργία της εφαρμογής υλοποιήθηκαν τα παρακάτω utilities:

- hideSoftKeyboard: χρησιμοποιείται για την απόκρυψη του πληκτρολογίου όταν ο χρήστης πατήσει το κουμπί αποστολής ή αλλαγής(ονόματος, προεπιλεγμένου μηνύματος). Θα μπορούσε να υλοποιηθεί και το αντίστοιχο για να το εμφανίζει, αλλά δε χρειάζεται καθώς το Android από μόνο του το εμφανίζει όταν πατηθεί κάποιο text area.
- isNetworkConnected: ελέγχει για το αν η συσκευή είναι συνδεδεμένη στο network.
- isInternetAvailable: στη περίπτωση που η συσκευή είναι διαθέσιμη στο network, κάνει έλεγχο για το αν είναι διαθέσιμη και στο internet.
- postRequest: δέχεται ένα url από κάποιον client μαζί με τα δεδομένα που πρέπει να στείλει στο backend και αναλαμβάνει να τα στείλει μέσω Post TCP/IP κλήσης. Για να πραγματοποιηθεί η κλήση πρέπει στο Manifest να έχει δοθεί άδεια χρήσης Internet.
- getRequest: δέχεται ένα url από κάποιον client μαζί με τα δεδομένα που πρέπει να στείλει στο backend και αναλαμβάνει να τα στείλει μέσω Get TCP/IP κλήσης. Για να πραγματοποιηθεί η κλήση πρέπει στο Manifest να έχει δοθεί άδεια χρήσης Internet.
- getUserLocation: παίρνει τη τοποθεσία του χρήστη από το GPS. Για να πραγματοποιηθεί με επιτυχία, πρέπει στο Manifest να έχει δοθεί άδεια καταγραφής της τοποθεσίας της συσκευής.
- saveData: παίρνει ως παράμετρο ένα key και ένα value και αποθηκεύει τοπικά στη συσκευή το value με αυτό το key. Χρησιμοποιείται για την αποθήκευση του ονόματος χρήστη και για το επιλεγμένο μήνυμα του καθενός.
- loadNameData: επιστρέφει το όνομα που είναι αποθηκευμένο στη συσκευή.
- loadMessageData: επιστρέφει το μήνυμα που είναι αποθηκευμένο στη συσκευή.

6.3.8. Resources

- drawable: ο φάκελος στον οποίο αποθηκεύονται όλες οι εικόνες που χρησιμοποιούνται στην εφαρμογή.
- layout: ο φάκελος στον οποίο δημιουργούνται τα XML αρχεία των Activities.
- menu: ο φάκελος που περιλαμβάνει τα XML αρχεία των αντικειμένων που βρίσκονται στη μπάρα κάθε Activity. Δημιουργήθηκε μόνο ένα, επειδή μόνο στο HomeActivity υπάρχει το Refresh button. Για τα υπόλοιπα, που έχουν το βελάκι Back, δεν χρειάζεται να φτιάξουμε αντίστοιχο XML αφού το Android το βάζει αυτόματα σε όλα.

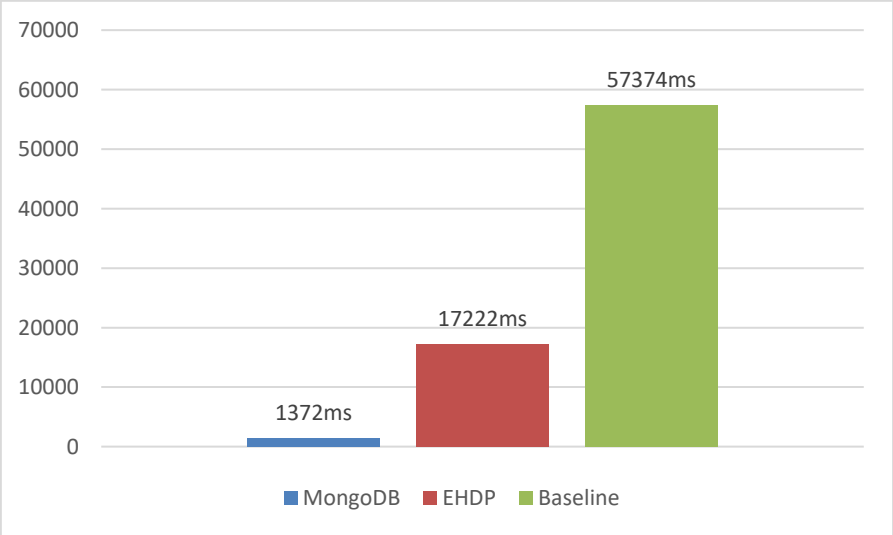
- mipmap: είναι ο φάκελος στον οποίο αποθηκεύεται η εικόνα της εφαρμογής. Υπάρχουν πολλοί φάκελοι mipmap ανάλογα την ανάλυση της συσκευής. Συνήθως είναι οι mipmap-hdpi, mipmap-mdpi, mipmap-xhdpi, mipmap-xxhdpi και mipmap-xxxhdpi.
- values: είναι ο φάκελος στον οποίο αποθηκεύονται τα XML αρχεία που παρέχουν κάποιες άλλες χρήσιμες πληροφορίες.
    - colors: το XML αρχείο στο οποίο δηλώνονται τα χρώματα που χρησιμοποιεί η εφαρμογή.
    - config: το XML αρχείο στο οποίο δηλώνονται κάποιες static final μεταβλητές.
    - strings: το XML αρχείο στο οποίο δηλώνονται κάποιες static final μεταβλητές.
    - styles: το XML αρχείο στο οποίο δηλώνεται η διάταξη των θεμάτων που χρησιμοποιούν τα Activities.

ΚΕΦ.7: Αποτελέσματα

Όπως περιγράφηκε στην ενότητα 3, για τον σκοπό της παρούσας πτυχιακής έγιναν πειραματικές δοκιμές σε αλγόριθμους τοποθέτησης δεδομένων στη κατανεμημένη βάση MongoDB. Σε αυτή την ενότητα παρουσιάζονται τα αποτελέσματα των πειραμάτων. Για την ακρίβεια των αποτελεσμάτων, κάθε αλγόριθμος εξετάστηκε πολλές φορές και με διάφορα σενάρια χρήσης. Το τελικό αποτέλεσμα, κάθε πειράματος, προέκυψε από τον μέσο όρο των αποτελεσμάτων των δοκιμών. Στο τέλος συγκρίνονται μεταξύ τους τα αποτελέσματα ώστε να δειχθεί ο αποδοτικότερος. Το κριτήριο για την απόδοση του συστήματος αποτελεί ο χρόνος εκτέλεσης του ερωτήματος.
Σε όλα τα ερωτήματα ο χρόνος εκτέλεσης μετριέται σε millisecond.
Στα παρακάτω διαγράμματα παρουσιάζονται οι μέσοι όροι των μετρήσεων.

7.1. Insert query

Στο παρακάτω διάγραμμα παρουσιάζεται ο χρόνος που απαιτείται για την τοποθέτηση 1000 εγγραφών(users) στη βάση, ανά αλγόριθμο.



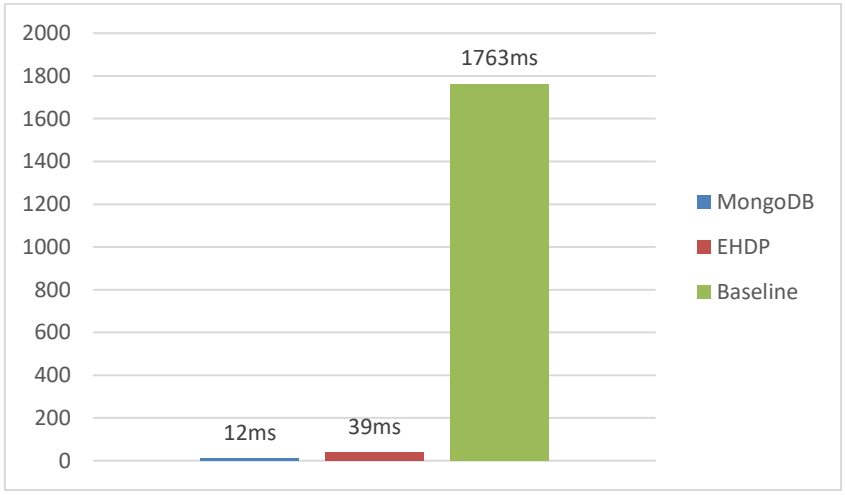
Διάγραμμα1: Insert query

Παρατηρούμε ότι η τοποθέτηση πολλών των αντικειμένων χρησιμοποιώντας την μέθοδο-αλγόριθμο που προσφέρει η MongoDB είναι 10 φορές πιο γρήγορη από την αντίστοιχη του EHDP, και περίπου 50 φορές πιο γρήγορη από την μέθοδο εισαγωγής ενός αντικειμένου χωρίς να λάβουμε υπόψιν ότι πρόκειται για γεωχωρικό. Ο αλγόριθμος της MongoDB όχι μόνο είναι ο πιο γρήγορος αλλά αποθηκεύει με σωστό τρόπο τα δεδομένα ταξινομώντας τα κάνοντας χρήση του 2dsphere ευρετηρίου.

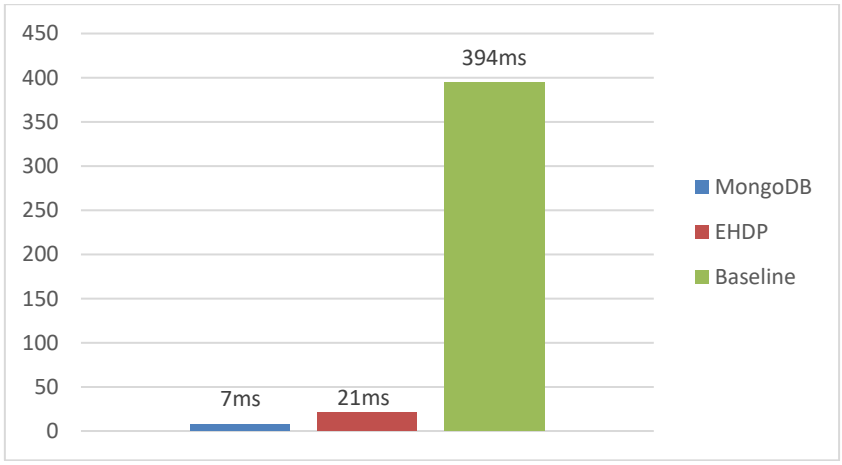
Όπως περιγράφηκε στο κεφάλαιο 3, δεν έχει σημασία ο χρόνος αποθήκευσης για την αξιολόγηση του αλγόριθμου παρά μόνο αν το response time δύο ή περισσότερων μοιάζει αρκετά.

7.2. Select query

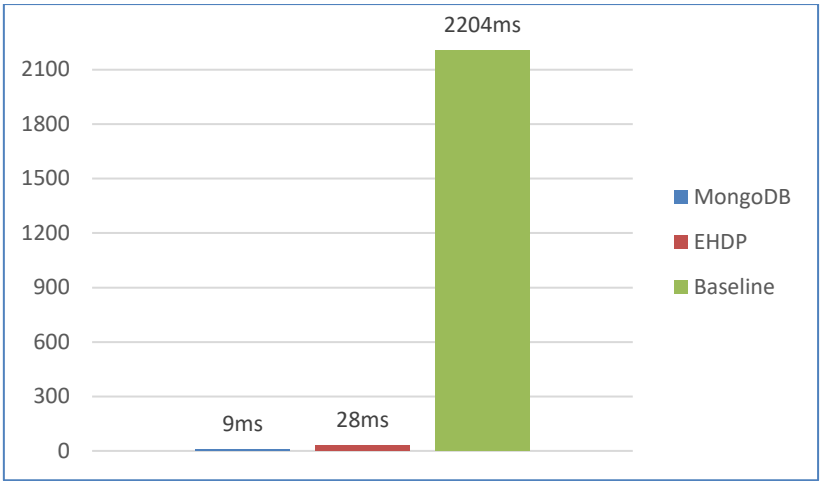
Στα παρακάτω διαγράμματα παρουσιάζονται οι χρόνοι απόκρισης της βάσης για την ανάκτηση των εγγραφών(users) ανάλογα την τοποθεσία τους , ανά αλγόριθμο, σε διάφορα σενάρια χρήσης.
Αναλυτικότερα, στο διάγραμμα 2, εμφανίζονται τα response time κάθε αλγόριθμου για την εύρεση όλων των χρηστών που είναι σε απόσταση μικρότερη ή ίση των 10 χιλιομέτρων από το σημείο αναφοράς.
Στο διάγραμμα 3, εμφανίζονται τα response time κάθε αλγόριθμου για την εύρεση όλων των χρηστών που είναι σε απόσταση μικρότερη ή ίση των 100 μέτρων από το σημείο αναφοράς.
Στο διάγραμμα 4, εμφανίζονται τα response time κάθε αλγόριθμου για την εύρεση των 30000 πιο κοντινών χρηστών από το σημείο αναφοράς.



Διάγραμμα2: Select nearest users within 10km query



Διάγραμμα3: Select nearest users within 100m query

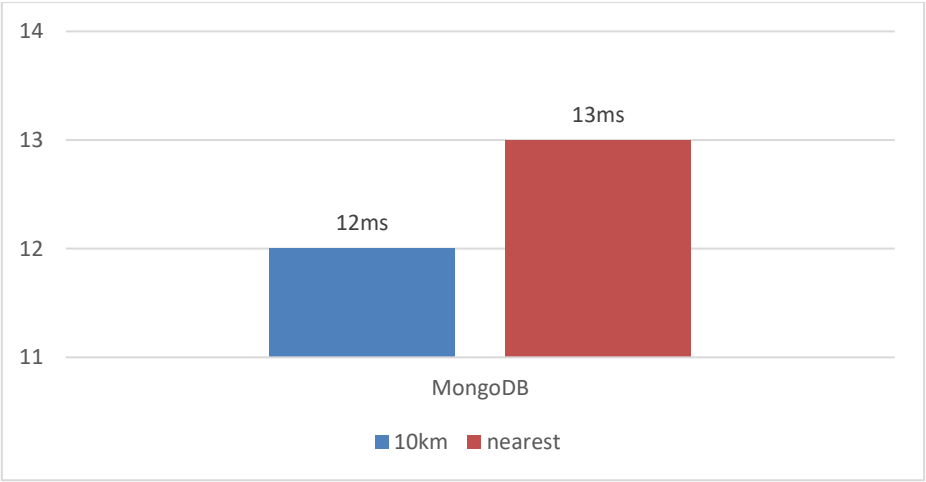


Διάγραμμα4: Select 30000 nearest users query

Παρατηρούμε ότι και στις τρεις περιπτώσεις, η διαφορά στους χρόνους απόκρισης των αλγορίθμων της MongoDB και του EHDP είναι σταθερή. Αυτός που χρησιμοποιείται από τη MongoDB είναι περίπου τρεις φορές πιο γρήγορος από τον EHDP, ωστόσο η διαφορά τους είναι αρκετά μικρή επειδή είναι λίγα milliseconds. Αντιθέτως, η μη χρησιμοποίηση κάποιας τεχνικής για την τοποθέτηση των αντικειμένων στη βάση έχει ως αποτέλεσμα αισθητή καθυστέρηση στον χρόνο επιστροφής των διαθέσιμων χρηστών.
Η διαφορά αυτή οφείλεται στον τρόπο που τοποθετήθηκαν τα αντικείμενα στη βάση, επειδή οι δύο πρώτοι αλγόριθμοι με τη χρήση του ευρετηρίου ομαδοποιούν τα κοντινά αντικείμενα και έτσι δε χρειάζεται να ανακαλυφθεί όλη η βάση για την εύρεση των πλησιέστερων χρηστών. Κάτι που δε γίνεται στη περίπτωση του τρίτου και έτσι πρέπει να γίνει αναζήτηση σε ολόκληρη τη βάση.

Ύστερα, έγινε άλλο ένα πείραμα για κάθε αλγόριθμο χωριστά. Σε αυτό το πείραμα συγκρίθηκαν τα response time της επιστροφής όλων των κοντινών χρηστών σε απόσταση μικρότερη ή ίση των 10 χιλιομέτρων με την επιστροφή του ίδιου αριθμού εγγραφών από τη βάση.

Για την περίπτωση του αλγόριθμου της MongoDB, που οι κοντινοί χρήστες σε απόσταση 10χιλιομέτρων από το σημείο αναφοράς είναι 60000, γίνεται σύγκριση με το response time του ίδιου αλγορίθμου για την επιστροφή των 60000 κοντινότερων χρηστών.



Διάγραμμα5: MongoDB

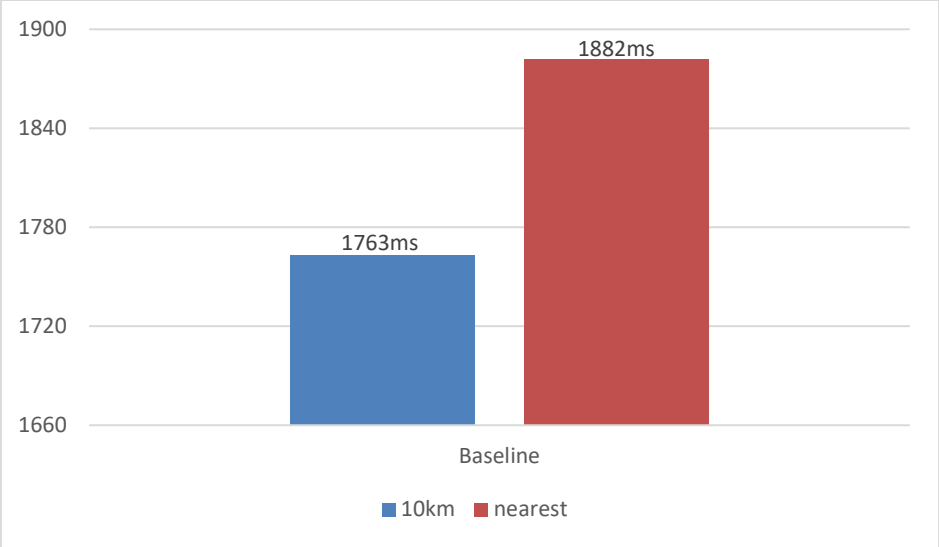
Βλέπουμε ότι η διαφορά στους χρόνους επιστροφής για το παραπάνω πείραμα είναι μόλις ένα millisecond, κάτι το οποίο θεωρείται αμελητέο και μπορούμε να πούμε ότι είναι ίδιος. Αυτό συμβαίνει επειδή με τον τρόπο που αποθηκεύτηκαν τα δεδομένα και λόγω της χρήσης του ευρετηρίου, είναι ταξινομημένα και έτσι ξέρει η βάση πότε πρέπει να σταματήσει την αναζήτηση.

Για την περίπτωση του αλγόριθμου EHDP, που οι κοντινοί χρήστες σε απόσταση 10χιλιομέτρων από το σημείο αναφοράς είναι 50000, γίνεται σύγκριση με το response time του ίδιου αλγορίθμου για την επιστροφή των 50000 κοντινότερων χρηστών.

Διάγραμμα6: EHDP

Όπως στην περίπτωση της MongoDB, έτσι και εδώ οι χρόνοι είναι οι ίδιοι. Και πάλι αυτό γίνεται λόγω σωστής τοποθέτησης των αντικειμένων στη βάση με τη χρήση ευρετηρίου.

Για την baseline υποδομή, που οι κοντινοί χρήστες σε απόσταση 10χιλιομέτρων από το σημείο αναφοράς είναι 20000, γίνεται σύγκριση με το response time του ίδιου αλγορίθμου για την επιστροφή των 20000 κοντινότερων χρηστών.

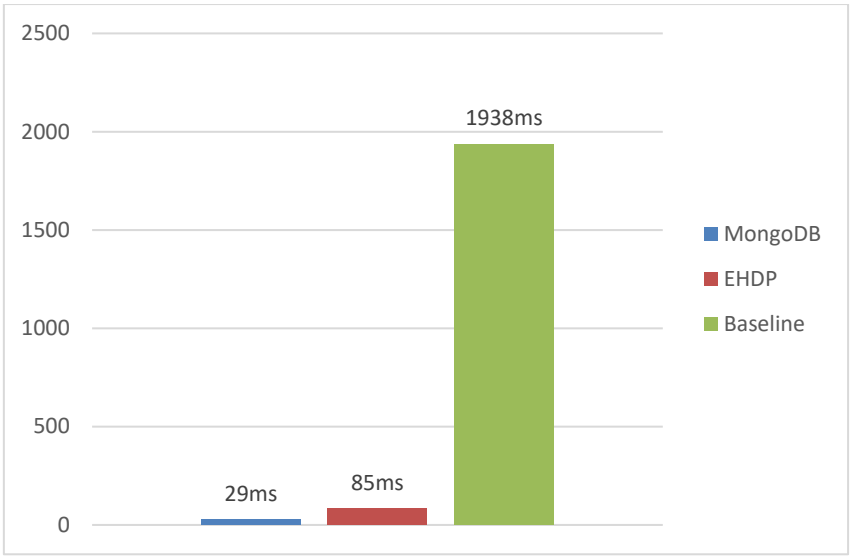


Διάγραμμα7: Baseline

Σε αντίθεση με τα προηγούμενα πειράματα, παρατηρείται αύξηση στο response time. Αυτό οφείλεται στο γεγονός ότι για την επιστροφή των 20000 κοντινότερων χρηστών ελέγχονται όλοι οι χρήστες μεταξύ τους ώστε να βρεθούν οι πιο κοντινοί, ενώ στην άλλη περίπτωση συγκρίνονται όλοι μόνο με το σημείο αναφοράς.

7.3. Select & Insert query

Στο παρακάτω διάγραμμα παρουσιάζεται ο χρόνος που απαιτείται για την τοποθέτηση 1 εγγραφής(user) στη βάση και ανάκτηση όλων των χρηστών που είναι κοντά του σε απόσταση 10 χιλιομέτρων, ανά αλγόριθμο. Για αυτό το σενάριο έγινε μόνο αυτή η δοκιμή επειδή όπως είδαμε ο χρόνος απόκρισης τόσο για απόσταση 10 χιλιομέτρων όσο και για την αντίστοιχη των 100 μέτρων είναι περίπου ο ίδιος, και επειδή πρόκειται για το σενάριο που χρησιμοποιείται στην εφαρμογή προτιμήθηκε η απόσταση των 10 χιλιομέτρων αφού επιστρέφει περισσότερα αποτελέσματα.



Διάγραμμα8: Insert & Select query

Παρατηρούμε ότι οι χρόνοι είναι ελάχιστα αυξημένοι με τους αντίστοιχους του προηγούμενου πειράματος. Αυτό οφείλεται στο γεγονός ότι υπάρχει η καθυστέρηση για την εγγραφή του χρήστη στη βάση.

7.4. Συμπεράσματα

Ο τρόπος που αποθηκεύονται τα αντικείμενα στη βάση αποτελεί σημαντικό παράγοντα στο χρόνο εκτέλεσης των ερωτημάτων που αφορούν γεωχωρικά δεδομένα. Ο πιο αποδοτικός τρόπος αποθήκευσής τους είναι η συσταδοποίηση των αντικειμένων που είναι σε σχετικά μικρή απόσταση μεταξύ τους, ώστε να μη χρειάζεται η ανάγνωση ολόκληρης της βάσης για την εύρεση μόνο των πλησιέστερων αντικειμένων.

Μπορούμε εύκολα να διαπιστώσουμε, στις περιπτώσεις των αλγορίθμων της MongoDB και του EHDP, ότι οι χρόνοι εύρεσης των χρηστών που είναι σε μια συγκεκριμμένη εμβέλεια από το χρήστη είναι περίπου οι ίδιοι(υπάρχει μια μικρή αύξηση) ανεξάρτητα από την απόσταση. Επίσης, το σταθερό πλήθος αποτελεσμάτων δεν επηρεάζει το response time. Αυτό συμβαίνει επειδή γίνεται χρήση ευρετηρίων και έτσι τα αντικείμενα ομαδοποιούνται κατά την εισχώρησή τους στη βάση. Επιπλέον, στην περίπτωση της MongoDB, που γίνεται χρήση του $geoNear, τα αντικείμενα επιστρέφονται ήδη ταξινομημένα, οπότε δεν γίνεται κάποιος επιπλέον υπολογισμός ώστε να ταξινομηθούν.
Τέλος, στη baseline υποδομή, παρατηρείται ότι ο χρόνος απόκρισης εξαρτάται από το πλήθος των στοιχείων που επιστρέφονται, αλλά και από τον τρόπο επιστροφής αυτών. Όσο πιο πολλά αποτελέσματα επιστρέφονται, τόσο πιο μεγάλο είναι το response time. Επίσης, ο χρόνος αυξάνεται για την επιστροφή συγκεκριμένου αριθμού σε σχέση με τον αντίστοιχο επιστροφής για μια περιοχή που φέρνει τον ίδιο αριθμό αποτελεσμάτων.

Σύμφωνα με τα αποτελέσματα των πειραματικών δοκιμών, όπως παρουσιάστηκαν παραπάνω, βγάζουμε το συμπέρασμα ότι ο αλγόριθμος που εφαρμόζεται από τη MongoDB είναι ο πιο αποδοτικός. Ο αλγόριθμος EHDP, βελτιώνει σημαντικά την απόδοση του συστήματος συγκριτικά με την baseline υποδομή και είναι ανταγωνιστικός της MongoDB, αφού όπως μπορούμε να διακρίνουμε, επιστρέφει εξίσου γρήγορα μεγάλο όγκο δεδομένων. Με μια πιο προσεκτική ματιά, ο αλγόριθμος της MongoDB, επιστρέφει τα αποτελέσματα κατά μέσο όρο σε περίπου 15 millisecond, ενώ ο EHDP σε λιγότερο από 40.

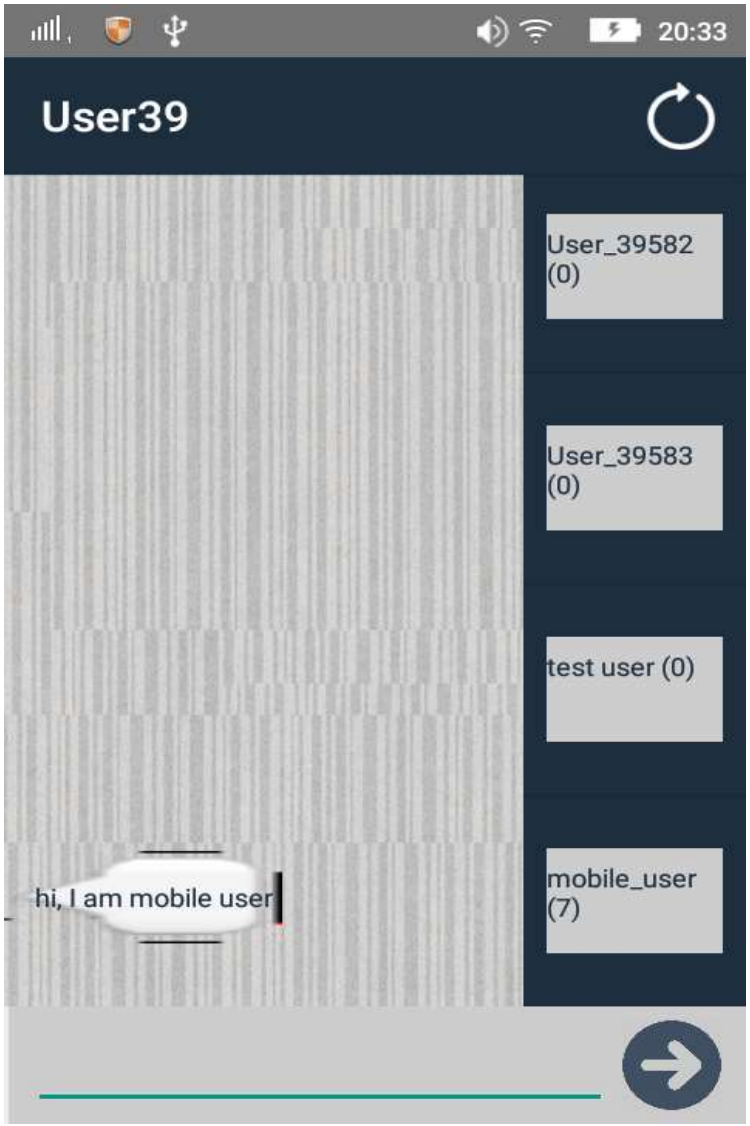
ΚΕΦ. 8: Εικόνες Εφαρμογής

8.1. Αρχική Οθόνη

Παρακάτω εμφανίζονται τα σενάρια εμφάνισης της αρχικής οθόνης.

8.1.1.

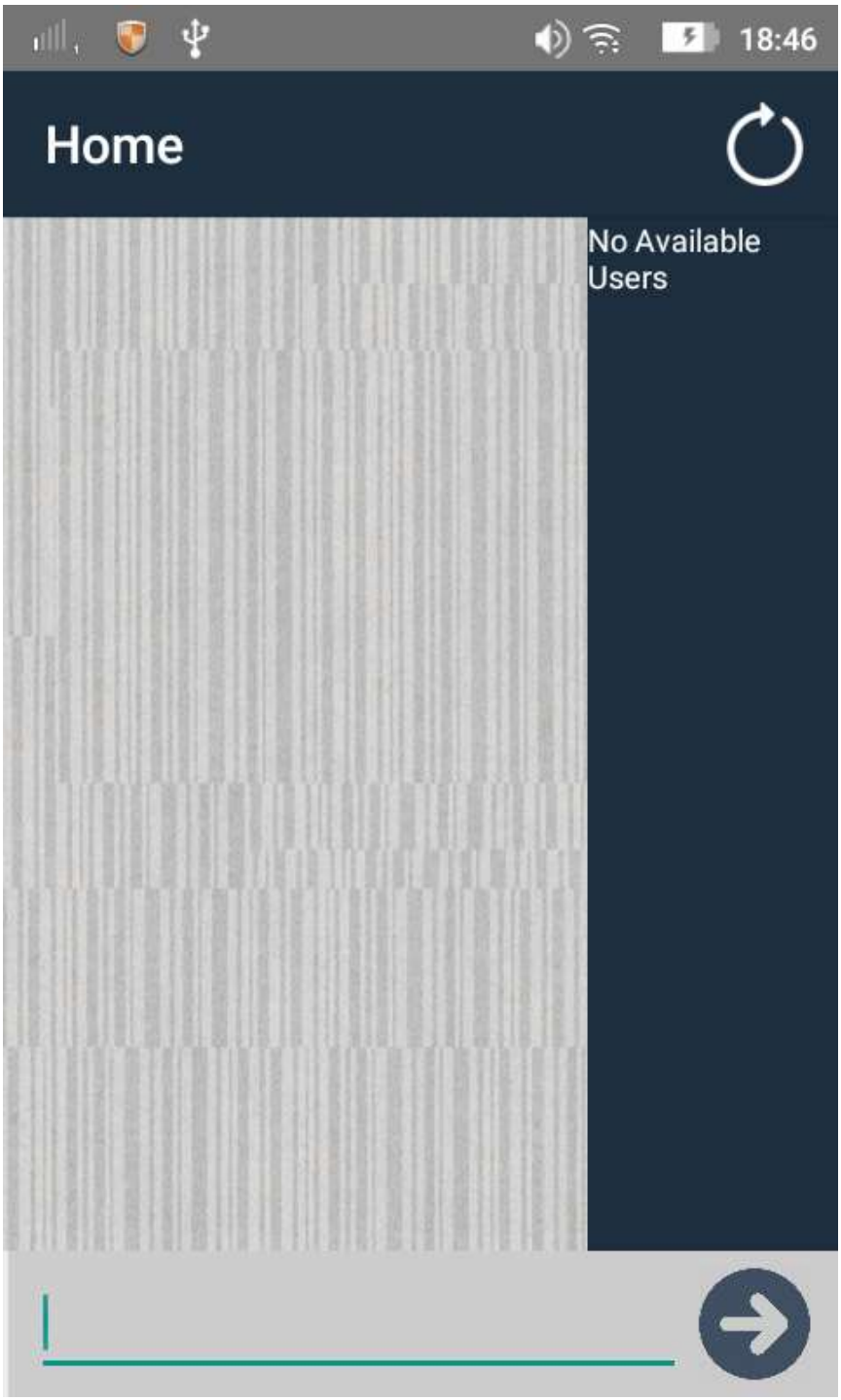
Η εικόνα 13 παρουσιάζει το πως είναι σε πλήρη μορφή, δηλαδή μετά την επιτυχημένη καταχώριση της τοποθεσίας του χρήστη στη βάση εμφανίζονται στη λίστα στα δεξιά της οθόνης οι κοντινοί του χρήστες που βρέθηκαν. Επιπλέον, στη λίστα των μηνυμάτων εμφανίζονται τα μηνύματα μεταξύ του χρήστη και του χρήστη που αναγράφεται στο πάνω μέρος της οθόνης.



Εικόνα 13: Home Activity

8.1.2

Η εικόνα 14 παρουσιάζει την πρώτη οθόνη της εφαρμογής όπως αυτή προκύπτει στη περίπτωση που μετά την επιτυχημένη και σωστή καταχώριση της τοποθεσίας του χρήστη στη βάση δεν βρεθεί κανένας κοντινός χρήστης.

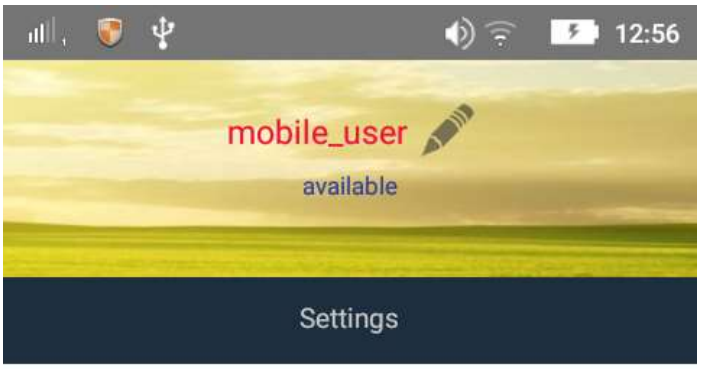


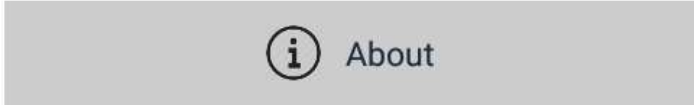
Εικόνα 14: No Available Users

Όπως διακρίνεται στην παραπάνω εικόνα, τα κύρια στοιχεία της οθόνης είναι ίδια με τη αντίστοιχη του προηγούμενου σεναρίου, αλλά η λίστα των μηνυμάτων είναι άδεια και αυτή των διαθέσιμων-κοντινών χρηστών εμφανίζει μήνυμα ενημέρωσης ότι δεν βρέθηκε κανένας χρήστης κοντά σε αυτόν.

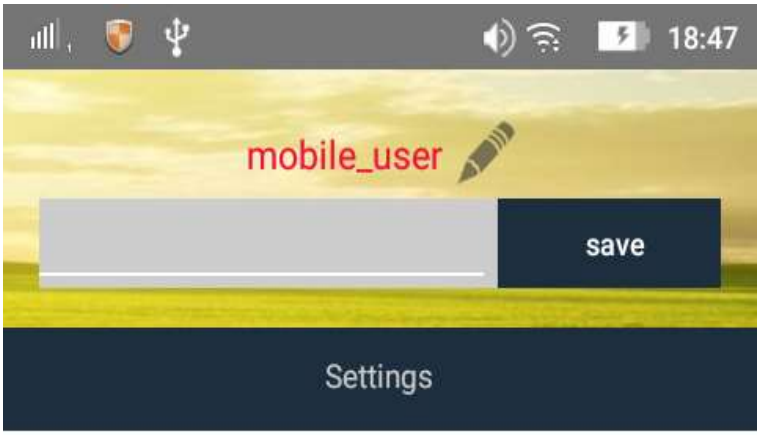
8.2. Profile Χρήστη

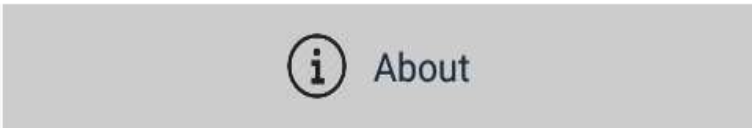
Παρακάτω εμφανίζονται τα σενάρια εμφάνισης της οθόνης που αποτελεί το profile του χρήστη. Όπως στην περίπτωση της αρχικής οθόνης, έτσι και σε αυτήν την οθόνη υπάρχουν δύο σενάρια χρήσης. Το πρώτο σενάριο είναι η κανονική προβολή του profile, ενώ το δεύτερο αφορά τη περίπτωση που ο χρήστης θελήσει να αλλάξει το όνομά του.





Εικόνα 15: User Profile





Εικόνα 16: Change Username

Όπως διακρίνεται στην παραπάνω εικόνα, όταν ο χρήστης θέλει να αλλάξει το όνομα χρήστη του η κατάσταση στην οποία βρίσκεται αντικαθίσταται από το text area που του δίνει τη δυνατότητα να κάνει την αλλαγή αυτή.

ΚΕΦ. 9: Μελλοντικές Επεκτάσεις

Στο πλαίσιο αυτής της πτυχιακής, δοκιμάστηκαν και συγκρίθηκαν μέθοδοι-αλγόριθμοι τοποθέτησης γεωχωρικών αντικειμένων σε μη σχεσιακές κατανεμημένες βάσεις δεδομένων. Για την μελέτη τους, δημιουργήθηκε μια απλή Android εφαρμογή η οποία επιτρέπει την ανταλλαγή μηνυμάτων μεταξύ κοντινών χρηστών. Επειδή η εφαρμογή είναι αρκετά απλή έχει ευκαιρίες ανάπτυξης.

Πιο συγκεκριμένα, μερικές ενέργειες οι οποίες θα βελτίωναν την εφαρμογή είναι:

- δοκιμή και άλλων spatial placement αλγόριθμων.
- καταχώριση της τοποθεσίας του χρήστη ενώ κινείται και ο υπολογισμός των διαθέσιμων προς αυτόν χρηστών σε πραγματικό χρόνο.
- διατήρηση των τοποθεσιών που επισκέφθηκε κάθε χρήστης στη βάση, έτσι ώστε στις πολυσύχναστες τοποθεσίες του να βρίσκει και άλλους χρήστες που την έχουν επισκεφθεί πολλές φορές.
- η αποθήκευση στατιστικών επικοινωνίας χρηστών, και ο υπολογισμός της απόστασης για χρήστες που έχουν πολλές συνομιλίες μεταξύ τους.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Online Available: https://www.digitalocean.com/community/tutorials/an-introduction-to-big-data-concepts-and-terminology

[2] French, Carl (1996). Data Processing and Information Technology(10th ed.). Thomson. p. 2. ISBN 1844801004.

[3] E.Pourabbas, “Geographical Information Systems, Trends and Techonologies”, pp.73-77 ,2014

[4] L.Einav, J.Levin, “Economics in the age of big data”, Vol.346 Issue 6210, 2014

[5] Online Available http://www.nbcnews.com/politics/elections/how-big-data-broke-american-politics-n732901

[6] Online Available: https://scholar.google.com/intl/us/scholar/help.html#coverage

[7] About the size of Google Scholar: playing the numbers. Granada: EC3 Working Papers, 18: 23 July 2014.

[8] B.Srinivasulu, A.Mebrahtu,”Concepts and Technologies of Big Data Management and Hadoop File System”, International Journal of Computer Trends and Technology, Vol.44 Issue 2, pp.80-88, 2017

[9] S.Khan, X.Liu, K.Shakil, M.Alam, “A survey on scholarly data: From big data perspective”,Information Processing & Management, Vol.53 Issue 4, pp.923-944, 2017

[10] J.Yoon, D.Jeong, C.Kang, S.Lee, “Forensic investigation framework for the document store NoSQL DBMS: MongoDB as a case study”, Digital Investigation, Vol.17, pp.53-65, 2016

[11] Online Available: https://cloud.google.com/bigtable/

[12] Online Available: http://cassandra.apache.org/

[13] Online Available: http://couchdb.apache.org/#about

[14] D.Abadi, P.Boncz, S.Harizopoulos, S.Idreos, S.Madden, “The Design and Implementation of Modern Column-Oriented Database Systems”, Foundations and Trends in Databases, Vol.5 Issue 3, pp.197-280, 2012

[15] D.Karger, E.Lehman, T.Leighton, R.Panigrahy, M.Levine, D.Lewin, "Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. Proceedings of the twenty-ninth annual ACM symposium on Theory of computing", pp.654-663, 1997

[16] DISTRIBUTED ALGORITHMS IN NOSQL DATABASES https://highlyscalable.wordpress.com/2012/09/18/distributed-algorithms-in-nosql-databases/

[17] Paiva J, Rodrigues L. On data placement in distributed systems. ACM SIGOPS Operating Systems Review. 2015;49(1):126-30.

[18] Online Available https://docs.mongodb.com/manual/applications/geospatial-indexes/#location-data

[19] X.Fubiao, C.Chengqi, C.Dong, D.Fang, “An efficient hierarchical data placement algorithm for massive spatial data storage systems”, Institute of Remote Sensing and GIS, pp.612-615, 2013

[20] Online Available: http://geojson.org/geojson-spec.html#geojson-objects
[21] Online Available: http://geojson.org/geojson-spec.html#point
[22] Online Available https://www.mongodb.com/blog/post/geospatial-performance-improvements-in-mongodb-3-2
[23] B.Gabrys, A.Bargiela, "General fuzzy min-max neural network for clustering and classification", Vol.11 Issue 3, pp.769-783, 2000
[24] W.Litwin, MA.Neimat, DA. Schneider LH*-A scalable, distributed data structure. ACM Trans on Database System, pp.480-525 , 1996
[25] Brinkman, K.Salzwedel, C.Scheideler, "Compact, adaptive placement schemes for non-uniform distribution requirements.", Proc of the 14th ACM Symp on Parallel Algorithms and Architectures. , pp.53-62, 2002
[26] C.Tao, X.Nong, L.Fang, "An Efficient Hierarchical Object Placement Algorithm for Object Storage Systems", Vol.49 Issue 4, pp.887-899, 2012
[27] Swift BP. Data placement in a scalable transactional data store. Master's thesis, Vrije Universiteit, Amsterdam, Netherland. 2012